

# Création de pages Web Dynamiques Côté serveur (en PHP)

Hicham EL KADIRI

## Introduction(1)

### ■ Internet et les pages web

- HTML : conception de pages destinées à être publiées sur Internet
- Page html : contient le texte à afficher et des instructions de mise en page
- HTML est un langage de description de page et non pas un langage de programmation
  - pas d'instructions de calcul ou pour faire des traitements suivant des conditions
- Des sites de plus en plus riches en informations
  - Nécessité croissante d'améliorer le contenu de sites
  - Mises à jour manuelles trop complexes
    - Pourquoi ne pas automatiser les mises à jour ?

## Introduction(2)

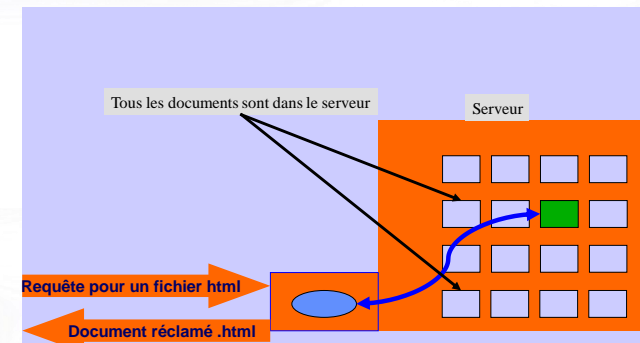
### ■ Pages web statiques : fonctionnement

- Leurs contenus ne changent ni en fonction du demandeur ni en fonction d'autres paramètres éventuellement inclus dans la requête adressée au serveur. Toujours le même résultat.
  - Rôle du serveur : localiser le fichier correspondant au document demandé et répond au navigateur en lui envoyant le contenu de ce fichier

### ■ Pages web statiques : limites

- Besoin de réponses spécifiques : passage de pages statiques à pages dynamiques

## Introduction(3)

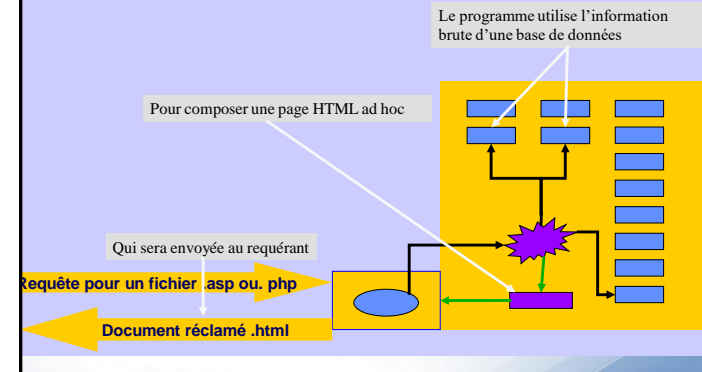


## Introduction(4)

### ■ Les langages de script-serveur : Définition

- Un langage de script -serveur est :
  - un programme stocké sur un serveur et exécuté par celui-ci,
  - qui passe en revue les lignes d'un fichier source pour en modifier une partie du contenu,
  - avant de renvoyer à l'appelant ( un navigateur par exemple) le résultat du traitement.
- La tâche d'interprétation des ordres à exécuter est déléguée à un composant, souvent appelé moteur,
  - installé sur le serveur,
  - qui est doté d'une API et d'un fonctionnement identique quel que soit la plate-forme utilisée pour gérer le serveur

## Introduction(5)



## Introduction(6)

### ■ Pages web dynamiques côté serveur ou côté client

- **Langage côté client** : traité par la machine qui accueille le logiciel de navigation.
  - Ses résultats peuvent varier en fonction de plate-forme utilisée. Un programme en JavaScript pourra fonctionner sous FireFox et poser problème sous Internet explorer.
  - Les résultats peuvent être différents suivant la machine (PC, Mac)
  - Nécessité de tests importants
  - Ne permettent pas de masquer les sources du programme
  - Sont indépendants du serveur et donc de l'hébergement

## Introduction(7)

### ■ Pages web dynamiques côté serveur ou côté client

- **Langage côté serveur** : le travail d'interprétation du programme est réalisé par le serveur
  - Sont indépendants de la machine et du logiciel de navigation utilisés pour la consultation.
  - Sont compatibles avec tous les navigateurs et toutes leurs versions.
  - Permettent de masquer les sources de ses programmes
  - Nécessitent de recharger la page chaque fois que celle-ci est modifiée.
- Pages côté serveur et côté client :
  - Script côté client pour des calculs et des traitements simples
  - Scripts côté serveur pour des calculs, des traitements et des mises à jours plus conséquents

## Introduction(8)

### ■ Les langages de création de pages web dynamiques côté serveur

#### • Les CGI

- Sont des composants exécutables (fichiers .exe ou .dll) qui produisent sur le serveur des contenus html à envoyer aux clients.
- Les CGI sont compilés. Ils sont rapides mais fortement liés à la plate-forme sur laquelle ils tournent.

#### ➢ PERL

- Surcharge rapide du serveur par la création de plusieurs processus
- Employé sur de nombreux serveurs. Il tourne sur de nombreuses plateformes : Unix, Linux, Windows, Mac
- Prévu à l'origine pour la manipulation de chaînes de caractères, il est rapidement devenu un véritable langage orienté objet.
- Abord difficile et faible lisibilité.

## Introduction(9)

### ■ Les langages de création de pages web dynamiques côté serveur

#### • ASP

- Basé sur des scripts écrits en VBScript, Jscript ou Javascript.
- Largement répandu,
- Facilité de mise en œuvre
- Plusieurs outils de développement intégrés (Macromédia Ultradev, Microsoft Visual Interdev).
- Intimement liée à l'environnement Windows NT/2000 et au serveur IIS (Internet Information Server) de Microsoft.
- L'environnement Microsoft est nécessaire

## Introduction(10)

### ■ Les langages de création de pages web dynamiques côté serveur

#### • JSP

- Constitue la réponse de Sun aux ASP de Microsoft
- Utilisation de Java
  - Au départ simple extension du langage Java
  - Est devenu un véritable langage de développement web
- Possède une interface de qualité
- Lenteur relative

## Introduction(11)

### ■ Les langages de création de page web dynamiques côté serveur

#### • PHP

- Connaît un succès toujours croissant sur le Web et se positionne comme un rival important pour ASP
- L'environnement Linux est sa plateforme de prédilection
- Combiné avec le serveur Web Apache et la base de données MySQL, PHP offre une solution particulièrement robuste, stable et efficace
- Gratuité : Tous les logiciels sont issus du monde des logiciels libres (Open Source).

## Un peu d'histoire

### ■ Histoire et Origine

- PHP : Hypertext PreProcessor
- Première version de PHP a été mis au point au début d'automne par Rasmus Lerdorf en 1994
  - Version appelée à l'époque Personal Home Pages
  - Pour 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
- La version 3.0 de PHP fut disponible le 6 juin 1998
- A la fin de l'année 1999, une version bêta de PHP, baptisée PHP4 est apparue
- En 2001 cinq millions de domaines utilisent PHP
  - trois fois plus que l'année 2000

## PHP : C'est QUOI ?

### ■ Définition

- Un langage de scripts permettant la création d'applications Web
- Indépendant de la plate-forme utilisée puisqu'il est exécuté côté serveur et non côté client.
- La syntaxe du langage provient de celles du langage C, du Perl et de Java.
- Ses principaux atouts sont:
  - La gratuité et la disponibilité du code source
  - La simplicité d'écriture de scripts
  - La possibilité d'inclure le script PHP au sein d'une page HTML
  - La simplicité d'interfaçage avec des bases de données
  - L'intégration au sein de nombreux serveurs web (Apache, Microsoft IIS, ...)

## Intégration PHP et HTML (1)

### ■ Principe

- Les scripts PHP sont généralement intégrés dans le code d'un document HTML
- L'intégration nécessite l'utilisation de balises
  - Avec le style php: `<?php ligne de code PHP ?>`

## Intégration PHP et HTML (2)

### ■ Forme d'une page PHP

- Intégration directe

```
< ?php
//ligne de code PHP
?>
<html>
<head> <title> Mon script PHP
</title> </head>
<body>
//ligne de code HTML
< ?php
//ligne de code PHP
?>
//ligne de code HTML
...
</body> </html>
```

## Intégration PHP et HTML (3)

- Forme d'une page PHP
  - Inclure un fichier PHP dans un fichier HTML : include

Fichier Principal

```
<html>
<head>
<title> Fichier d'appel </title>
</head>
<body>
<?php
$salut = " BONJOUR" ;
include "information_inc.php" ;
?>
</body>
</html>
```

Fichier à inclure :  
information\_inc.php

```
<?php
$chaine=$salut. ' , C'est PHP ' ;
//Ou $chaine=$salut. " , C'est PHP " ;
echo " <table border= '3'>
<tr> <td width = '100%'>
<h2> $chaine</h2>
</td> </tr></table>" ;
?>
```

BONJOUR , C'est PHP

## Intégration PHP et HTML (4)

### ■ Envoi du code HTML par PHP

- La fonction echo : `echo Expression;`
  - `>echo "Chaine de caracteres";`
  - `>echo (1+2)*87;`
- La fonction print : `print(expression);`
  - `>print("Chaine de caracteres");`
  - `>print ((1+2)*87);`
- La fonction printf : `printf (chaîne formatée);`
  - `>printf ("Le périmètre du cercle est %d",$Perimetre);`

## Intégration PHP et HTML (5)

```
<?php
$n = 43951789;
$u = -43951789;
$c = 65; // ASCII 65 is 'A'

// notez le double %, cela affiche un caractère '%' littéral
printf("%%b = %b\n", $n); // représentation binaire
printf("%%c = %c\n", $c); // affiche le caractère ascii, comme la fonction chr()
printf("%%d = %d\n", $n); // représentation standard d'un entier
printf("%%e = %e\n", $n); // notation scientifique
printf("%%u = %u\n", $n); // représentation entière non signée d'un entier positif
printf("%%u = %u\n", $u); // représentation entière non signée d'un entier négatif
printf("%%f = %f\n", $n); // représentation en virgule flottante
printf("%%o = %o\n", $n); // représentation octale
printf("%%s = %s\n", $n); // représentation chaîne de caractères
printf("%%x = %x\n", $n); // représentation hexadécimal (minuscule)
printf("%%X = %X\n", $n); // représentation hexadécimal (majuscule)

printf("%%+d = %+d\n", $n); // indication du signe pour un entier positif
printf("%%+d = %+d\n", $u); // indication du signe pour un entier négatif
?>
```

## Syntaxe de base : Introduction

### ■ Typologie

- Toute instruction se termine par un point-virgule
- Sensible à la casse
  - Sauf par rapport aux fonctions

### ■ Les commentaires

- `/* Voici un commentaire! */`
- `// un commentaire sur une ligne`

## Syntaxe de base : Les constantes

### ■ Les constantes

- Define("nom\_constante", valeur\_constante )
  - define ("ma\_const", "Vive PHP");
  - define ("an", 2022);
- Les constantes prédéfinies
  - NULL
  - \_FILE\_
  - \_LINE\_
  - PHP\_VERSION
  - PHP\_OS
  - TRUE et FALSE
  - E\_ERROR

## Syntaxe de base : Les variables (1)

- Principe
  - Commencent par le caractère \$
  - N'ont pas besoin d'être déclarées
- Fonctions de vérifications de variables
  - Doubleval(convertit une chaîne en nombre à virgule flottante), empty(Détermine si une variable est vide), gettype(Retourne le type de la variable), intval(Retourne la valeur numérique entière),
  - is\_array(Détermine si une variable est un tableau), is\_bool(Détermine si une variable est un booléen), is\_double(Détermine si une variable est de type nombre décimal), is\_float(), is\_int(), is\_integer, is\_long(), is\_object(étermine si une variable est de type objet), is\_real(), is\_numeric(), is\_string()
  - isset(détermine si une variable est déclarée et est différente de null), settype(Affecte un type à une variable), strval(Récupère la valeur d'une variable, au format chaîne), unset(Détruit une variable)
- Affectation par valeur et par référence
  - Affectation par valeur : \$b=\$a
  - Affectation par (référence) variable : \$c = &\$a

## Syntaxe de base : Les variables(2)

### ■ Visibilité des variables

- Variable locale
  - Visible uniquement à l'intérieur d'un contexte d'utilisation
- Variable globale
  - Visible dans tout le script
  - Utilisation de l'instruction global dans des contextes locales

```
<?php
$a = 1;
$b = 2;
function somme() {
    global $a, $b;
    $b = $a + $b;
}
somme();
echo $b; //3
```

En déclarant globales les variables \$a et \$b locales de la fonction somme(), toutes les références à ces variables concerneront les variables globales.

## Syntaxe de base : Les variables(3)

### ■ Les variables dynamiques

- Permettent d'affecter un nom différent à une autre variable
 

```
$nom_variable = 'nom_var';
${$nom_variable} = valeur; // équivaut à $nom_var = valeur;
```
- Les variables tableaux sont également capables de supporter les noms dynamiques
 

```
$nom_variable = array("val0", "val1", ..., "valN");
${$nom_variable[0]} = valeur; // équivaut à $val0 = valeur;
$nom_variable = "nom_var";
${$nom_variable}[0] = valeur; // équivaut à $nom_var[0] = valeur;
```
- Les accolades servent aussi à éviter toute confusion lors du rendu d'une variable dynamique
 

```
echo "Nom : $nom_variable - Valeur : ${$nom_variable}";
//équivaut à echo "Nom : $nom_variable - Valeur : $nom_var";
```

## Syntaxe de base : Les variables (4)

### ■ Variables prédéfinies

- Les variables d'environnement du serveur dépendant du client

Variable	Description
<code>\$_SERVER["HTTP_HOST"]</code>	Nom d'hôte de la machine du client (associée à l'adresse IP)
<code>\$_SERVER["HTTP_REFERER"]</code>	URL de la page qui a appelé le script PHP
<code>\$_SERVER["HTTP_ACCEPT_LANGUAGE"]</code>	Langue utilisée par le serveur (par défaut en-us)
<code>\$_SERVER["HTTP_ACCEPT"]</code>	Types MIME reconnus par le serveur (séparés par des virgules)
<code>\$_SERVER["CONTENT_TYPE"]</code>	Type de données contenu présent dans le corps de la requête. Il s'agit du type MIME des données
<code>\$_SERVER["REMOTE_ADDR"]</code>	L'adresse IP du client appelant le script CGI
<code>\$_SERVER["PHP_SELF"]</code>	Nom du script PHP

## Syntaxe de base : Les variables (5)

### ■ Variables prédéfinies

- Les variables d'environnement dépendant du serveur

Variable	Description
<code>\$_SERVER["SERVER_NAME"]</code>	Le nom du serveur
<code>\$_SERVER["HTTP_HOST"]</code>	Nom de domaine du serveur
<code>\$_SERVER["SERVER_ADDR"]</code>	Adresse IP du serveur
<code>\$_SERVER["SERVER_PROTOCOL"]</code>	Nom et version du protocole utilisé pour envoyer la requête au script PHP
<code>\$_SERVER["DATE_GMT"]</code>	Date actuelle au format GMT
<code>\$_SERVER["DATE_LOCAL"]</code>	Date actuelle au format local
<code>\$_SERVER["\$DOCUMENT_ROOT"]</code>	Racine des documents Web sur le serveur

## Syntaxe de base : Les variables (6)

### ■ Variables prédéfinies

- Affichage des variables d'environnement

-la fonction `phpinfo()`

• `<?php phpinfo(); ?>`

• `echo phpinfo(constante);`

<code>INFO_CONFIGURATION</code>	affiche les informations de configuration.
<code>INFO_CREDITS</code>	affiche les informations sur les auteurs du module PHP
<code>INFO_ENVIRONMENT</code>	affiche les variables d'environnement.
<code>INFO_GENERAL</code>	affiche les informations sur la version de PHP.
<code>INFO_LICENSE</code>	affiche la licence GNU Public
<code>INFO_MODULES</code>	affiche les informations sur les modules associés à PHP
<code>INFO_VARIABLES</code>	affiche les variables PHP prédéfinies.

-la fonction `getenv()` (retourne la valeur d'une variable d'environnement)

• `<?php echo getenv("HTTP_USER_AGENT"); ?>`

## Syntaxe de base : Les types de données

### ■ Principe

- Pas besoin d'affecter un type à une variable avant de l'utiliser
- La même variable peut changer de type en cours de script
- Les variables issues de l'envoi des données d'un formulaire sont du type string

### ■ Les différents types de données

- Les entiers : le type **Integer**
- Les flottants : le type **Double**
- Les tableaux : le type **array**
- Les chaînes de caractères : le type **string**
- Les objets



## Syntaxe de base : Les types de données (2)

### ■ Le transtypage

- La fonction **settype()** permet de convertir le type auquel appartient une variable

```
<?php $nbre=10;
echo "la variable $nbre avant conversion est de type ". gettype($nbre);
settype($nbre, "double");
echo '<br>';
echo "la variable $nbre après conversion est de type ". gettype($nbre); ?>
```

**la variable 10 avant conversion est de type integer**  
**la variable 10 après conversion est de type double**

### ■ Détermination du type de données

- gettype(), is\_long(), is\_double(), is\_string(), is\_array(), is\_object(), is\_bool()

## Syntaxe de base : Les chaînes de caractères(1)

### ■ Principe

- Peuvent être constituées de n'importe quel caractère alphanumérique et de ponctuation, y compris les caractères spéciaux

```
\t La nouvelle monnaie unique, 1' Euro, est enfin là...\n\r
```

- Une chaîne de caractères doit être toujours entourée par des guillemets simples (') ou doubles (")

" Ceci est une chaîne de caractères valide."

'Ceci est une chaîne de caractères valide.'

"Ceci est une chaîne de caractères invalide."

- Des caractères spéciaux à insérer directement dans le texte, permettent de créer directement certains effets comme des césures de lignes

Car	Code ASCII	Code hex	Description
\car			échappe un caractère spécifique.
" "	32	0x20	un espace simple.
\t	9	0x09	tabulation horizontale
\n	13	0x0D	nouvelle ligne
\r	10	0x0A	retour à chariot
\0	0	0x00	caractère NUL
\v	11	0x0B	tabulation verticale

## Syntaxe de base : Les chaînes de caractères(2)

### ■ Quelques fonctions de manipulation

```
chaîne_result = addslashes(chaîne);
```

ajoute des antislashes dans une chaîne

Retourne la chaîne après avoir échappé tous les caractères qui doivent l'être

```
$sstr = "Votre nom est-il O'reilly ?";
```

```
echo addslashes($sstr); // Affiche : Votre nom est-il O\'reilly ?
```

```
chaîne_result = chop(chaîne);
```

supprime les espaces blancs en fin de chaîne.

```
caractère = chr(nombre);
```

retourne un caractère en mode ASCII

```
chaîne_result = crypt(chaîne [, chaîne_code])
```

code une chaîne avec une base de codage. Hachage à sens unique (indéchiffrable)

```
echo expression_chaîne;
```

affiche à l'écran une ou plusieurs chaînes de caractères.

```
$tableau = explode(délimiteur, chaîne);
```

scinde une chaîne en fragments à l'aide d'un délimiteur et retourne un tableau.

## Syntaxe de base : les opérateurs (1)

### ■ Les opérateurs

- les opérateurs de calcul
- les opérateurs d'assignation
- les opérateurs d'incrément
- les opérateurs de comparaison
- les opérateurs logiques
- les opérateurs bit-à-bit
- les opérateurs de rotation de bit



## Syntaxe de base : Les opérateurs(2)

### ■ Les opérateurs de calcul

Opérateur	Dénomination	Effet	Exemple	Résultat
+	opérateur d'addition	Ajoute deux valeurs	\$x+3	10
-	opérateur soustraction	Soustrait deux valeurs	\$x-3	4
*	opérateur multiplication	Multiplie deux valeurs	\$x*3	21
/	plus: opérateur division	Divise deux valeurs	\$x/3	2.3333333
=	opérateur d'affectation	Affecte une valeur à une variable	\$x=3	Met la valeur 3 dans la variable \$x

## Syntaxe de base : Les opérateurs(3)

### ■ Les opérateurs d'assignation

Opérateur	Effet
+=	addition de deux valeurs et stocke le résultat dans la variable (à gauche)
-=	soustrait deux valeurs et stocke le résultat dans la variable
*=	multiplie deux valeurs et stocke le résultat dans la variable
/=	divise deux valeurs et stocke le résultat dans la variable
%=	donne le reste de la division de deux valeurs et stocke le résultat dans la variable
=	Effectue un OU logique entre deux valeurs et stocke le résultat dans la variable
&=	Effectue un ET logique entre deux valeurs et stocke le résultat dans la variable
.=	Concatène deux chaînes et stocke le résultat dans la variable

## Syntaxe de base : Les opérateurs(4)

### ■ Les opérateurs d'incrémentement

Opérateur	Dénomination	Effet	Syntaxe	Résultat (avec x valant 7)
++	Incrémentation	Augmente d'une unité la variable	\$x++	8
--	Décrémentation	Diminue d'une unité la variable	\$x--	6

### ■ Les opérateurs de comparaison

Opérateur	Dénomination	Effet	Exemple	Résultat
==	opérateur d'égalité	Compare deux valeurs et vérifie leur égalité	\$x==3	Retourne 1 si \$X est égal à 3 sinon 0
<	opérateur d'infériorité stricte	Vérifie qu'une variable est strictement inférieure à une valeur	\$x<3	Retourne 1 si \$X est inférieur à 3 sinon 0
<=	opérateur d'infériorité	Vérifie qu'une variable est inférieure ou égale à une valeur	\$x<=3	Retourne 1 si \$X est inférieur à 3 sinon 0
>	opérateur de supériorité stricte	Vérifie qu'une variable est strictement supérieure à une valeur	\$x>3	Retourne 1 si \$X est supérieur à 3 sinon 0
>=	opérateur de supériorité	Vérifie qu'une variable est supérieure ou égale à une valeur	\$x>=3	Retourne 1 si \$X est supérieur ou égal à 3 sinon 0
!=	opérateur de différence	Vérifie qu'une variable est différente d'une valeur	\$x!=3	Retourne 1 si \$X est différent de 3 sinon 0

## Syntaxe de base : Les opérateurs(5)

### ■ Les opérateurs logiques

Opérateur	Dénomination	Effet	Syntaxe
ou OR	OU logique	Vérifie qu'une des conditions est réalisée	((condition1)    (condition2))
&& ou AND	ET logique	Vérifie que toutes les conditions sont réalisées	((condition1) && (condition2))
^	OU exclusif	Opère du OU logique	((condition1) ^ (condition2))
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)	!(condition)

### ■ Les opérateurs bit-à-bit

Opérateur	Dénomination	Effet	Syntaxe	Résultat
&	ET bit-à-bit	Retourne 1 si les deux bits de même poids sont à 1	9 & 12 (001 & 110)	8 (000)
	OU bit-à-bit	Retourne 1 si l'un ou l'autre des deux bits de même poids est à 1 (ou les deux)	9   12 (001   110)	13 (110)
^	OU bit-à-bit	Retourne 1 si l'un des deux bits de même poids est à 1 (mais pas les deux)	9 ^ 12 (001 ^ 110)	5 (010)
~	Complément (NOT)	Retourne 1 si le bit est à 0 (et inversement)	~9 (~001)	6 (011)

## Syntaxe de base : Les opérateurs(6)

### ■ Les opérateurs de rotation de bit

Opérateur	Dénomination	Effet	Syntaxe	Résultat
$\ll$	Rotation à gauche	Décale les bits vers la gauche (multiplie par 2 à chaque décalage). Les zéros qui sortent à gauche sont perdus, tandis que des zéros sont insérés à droite	$6 \ll 1$ (110)	12 (1100)
$\gg$	Rotation à droite avec conservation du signe	Décale les bits vers la droite (divise par 2 à chaque décalage). Les zéros qui sortent à droite sont perdus, tandis que le bit non nul de poids le plus fort est recopié à gauche	$6 \gg 1$ (011)	3 (011)

### ■ Autres opérateurs

Opérateur	Dénomination	Effet	Syntaxe	Résultat
.	Concaténation	Joint deux chaînes bout à bout	"Bonjour"/"Au revoir"	"BonjourAu revoir"
\$	Réaffectation variable	Permet de définir une variable	\$MVariable=2	
>	Propriété d'un objet	Permet d'accéder aux données membres d'une classe	\$MObjet->Propriété	

## Syntaxe de base : Les opérateurs(7)

### ■ Les priorités

Priorité des opérateurs									
0	1	2	3	4	5	6	7	8	9
~	++	!	~	-					
*	/	%							
+	-								
<	<=	>							
=	!=								
&									
^									
&&									
?	:								
=	+=	-=	*=	/=	%=	<<=	>=	>>=	&=
^									
AND									
XOR									

## Syntaxe de base : Les instructions conditionnelles(1)

- L'instruction if
  - if (condition réalisée) { liste d'instructions }
- L'instruction if ... else
  - if (condition réalisée) {liste d'instructions}
  - else { autre série d'instructions }
- L'instruction if ... elseif ... Else
  - if (condition réalisée) {liste d'instructions}
  - elseif (autre condition ) {autre série d'instructions }
  - else (dernière condition réalisée) { série d'instructions }
- Opérateur ternaire
  - (condition) ? instruction si vrai : instruction si faux

## Syntaxe de base : Les instructions conditionnelles(2)

### ■ L'instruction switch

```
switch (Variable) {
case Valeur1: Liste d'instructions break;
case Valeur2: Liste d'instructions break;
case Valeurs...: Liste d'instructions break;
default: Liste d'instructions break;
}
```

## Syntaxe de base : Les instructions conditionnelles(3)

- La boucle for
  - for (\$i=1; \$i<6; \$i++) { echo "\$i<br />"; }
- La boucle while
  - while(condition) {bloc d'instructions ;}
  - while (condition) :Instruction1 ;Instruction2 ;  
.... endwhile ;
- La boucle do...while
  - do {bloc d'instructions ;}while(condition) ;
- La boucle foreach
  - foreach (\$tableau as \$valeur) {insts utilisant \$valeur ;}

une façon simple de parcourir des tableaux. foreach ne fonctionne que pour les tableaux et les objets,

## Syntaxe de base : Les fonctions(1)

### ■ Déclaration et appel d'une fonction

```
function nom_fonction($arg1, $arg2, ...$argn)
{
    déclaration des variables ;
    bloc d'instructions ;
    //fin du corps de la fonction
    return $resultat ;
}
```

### ■ Fonction avec nombre d'arguments inconnu

- **func\_num\_args()** : fournit le nombre d'arguments qui ont été passés lors de l'appel de la fonction
- **func\_get\_arg(\$i)** : retourne la valeur de la variable située à la position \$i dans la liste des arguments passés en paramètres.
  - Ces arguments sont numérotés à partir de 0

## Syntaxe de base : Les fonctions(2)

### ■ Fonction avec nombre d'arguments inconnu

```
<?php
function produit() {
    $nbarg = func_num_args();
    echo " nb arguments = ".$nbarg. "<br />"; // $nb arguments
    $prod=1 ;
    // la fonction produit a ici $nbarg arguments
    for ($i=0; $i <$nbarg; $i++)
    {
        $prod *= func_get_arg($i);
        echo " prod : ".$prod. "<br />";
    }
    return $prod;
}

echo "le produit est : ". produit (3, 77, 10, 5, 81, 9). "<br />";
?>
```

// affiche le resultat :

```
nb arguments = 6
prod : 3
prod : 231
prod : 2310
prod : 11550
prod : 935550
prod : 8419950
le produit est : 8419950
```

## Syntaxe de base : Les fonctions(3)

### ■ Passage de paramètre par référence

- Pour passer une variable par référence, il faut que son nom soit précédé du symbole & (exemple &\$a)

```
<?php
function dire_texte($qui, &$texte){ $texte = "Bienvenue $qui";}
$chaine = "Bonjour ";
dire_texte("cher phpeur", $chaine);
echo $chaine; // affiche "Bienvenue cher phpeur"
?>
```

### ■ L'appel récursif

- PHP admet les appels récursifs de fonctions

## Syntaxe de base : Les fonctions(4)

### ■ Appel dynamique de fonctions

- Exécuter une fonction dont le nom n'est pas forcément connu à l'avance par le programmeur du script
- L'appel dynamique d'une fonction s'effectue en suivant le nom d'une variable contenant le nom de la fonction par des parenthèses

```
<?php
$datejour = getdate(); // date et heure du timestamp
//récupération des heures et minutes actuelles
$heure = $datejour[hours];
$minute = $datejour[minutes];
var_dump($datejour); // Affiche les informations de la variable :
function bonjour() {
    global $heure;
    global $minute;
    echo "<b> Bonsoir à vous il est : ". $heure. " H ". $minute. "</b> <br />";
}
function bonsoir() {
    global $heure;
    global $minute;
    echo "<b> Bonjour à vous il est : ". $heure. " H ". $minute. "</b> <br />";
}
if ($heure > 17) {bonjour();} else bonsoir();
//appel dynamique de la fonction
?>
```

```
array (size=11)
  'seconds' => int 39
  'minutes' => int 38
  'hours' => int 5
  'yday' => int 22
  'wday' => int 3
  'mon' => int 2
  'year' => int 2023
  'yday' => int 52
  'weekday' => string 'Wednesday' (length=9)
  'month' => string 'February' (length=8)
  0 => int 1677044319
```

Bonjour à vous il est : 5 H 38

## Syntaxe de base : Les fonctions(5)

### ■ Variables locales et variables globales

- variables en PHP : global, static, local
- toute variable déclarée en dehors d'une fonction est globale
- utiliser une variable globale dans une fonction, l'instruction **global** suivie du nom de la variable
- Pour conserver la valeur acquise par une variable entre deux appels de la même fonction : l'instruction **static**.

➤ Les variables statiques restent locales à la fonction et ne sont pas réutilisables à l'extérieur.

```
<?php
function cumul ($prix) {
    static $cumul = 0;
    static $i = 1;
    echo "Total des achats $i = ";
    $cumul += $prix; $i++;
    return $cumul; }
echo cumul (175). "<br />";
echo cumul (65). "<br />";
echo cumul (69). "<br />"; ?>
```

Total des achats 1 = 175  
Total des achats 2 = 240  
Total des achats 3 = 309

## Syntaxe de base : Les tableaux(1)

### ■ Principe

- Création à l'aide de la fonction **array()**
- Uniquement des tableaux à une dimension
  - Les éléments d'un tableau peuvent pointer vers d'autres tableaux
- Les éléments d'un tableau peuvent appartenir à des types distincts
- L'index d'un tableau en PHP commence de 0
- Pas de limites supérieures pour les tableaux
- La fonction **count()** pour avoir le nombre d'éléments d'un tableau

## Syntaxe de base : Les tableaux(2)

### ■ Les tableaux indicés et les tableaux associatifs

- Tableau indicé
  - Accéder aux éléments par l'intermédiaire de numéros

```
$tableau[indice] = valeur;
$jour[3] = "Mercredi";
$note[0] = 20;

$tableau = array(valeur0, valeur1, ..., valeurN);
$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi",
    "Vendredi", "Samedi");
Ou : $jour = ["Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi",
    "Vendredi", "Samedi"];
$note = array(20, 15, 12.6, 17, 10, 20, 11, 18, 19);
Ou : $note = [20, 15, 12.6, 17, 10, 20, 11, 18, 19];
$variable = $tableau[indice];
$JJ = $jour[6]; // affecte "Samedi" à $JJ
echo $note[1] + $note[5];
```

## Syntaxe de base : Les tableaux(3)

### ■ Les tableaux indicés et les tableaux associatifs

➤ Tableau associatif (ou table de hachage)

-Les éléments sont référencés par des chaînes de caractères associatives en guise de nom: la clé d'index

```
$tableau["indice"] = valeur;
$jour["Dimanche"] = 7;
$jour["Mercredi"] = "Le jour des enfants";

$tableau = array(ind0 => val0, ind1 => val1,..., indN => valN);
$jour = array("Dimanche" => 1, "Lundi" => 2, "Mardi" => 3, "Mercredi" => 4, "Jeudi" => 5, "Vendredi" => 6, "Samedi" => 7);

$variable = $tableau["indice"];
$JJ = $jour["Vendredi"]; //affecte 6 à $JJ
echo $jour["Lundi"]; //retourne la valeur 2
```

## Syntaxe de base : Les tableaux(4)

### ■ Tableaux multidimensionnels

- Pas d'outils pour créer directement des tableaux multidimensionnels
- L'imbrication des tableaux est possible

```
$tab1 = array(val0, val1,..., valN);
$tab2 = array(val0, val1,..., valN);
// Création d'un tableau à deux dimensions
$tableau = array($tab1, $tab2);
$mois = array("Janvier", "Février", "Mars", "Avril", "Mai", "Juin", "Juillet", "Août", "Septembre", "Octobre", "Novembre", "Décembre");
$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi");
if (isset($mois, $jour)) {
    echo 'Cette variable existe, donc je peux l\'afficher.';
}
$element_date = array($mois, $jour);
$variable = $tableau[indice][indice];
$MM = $element_date[0][0]; //affecte "Janvier" à $MM
echo $element_date[1][5] . " 7 " . $element_date[0][2] . " 2002";
// retourne "Jeudi 7 Mars 2002"
```

## Syntaxe de base : Les tableaux(5)

### ■ Lecture des éléments d'un tableau

- Avec une boucle for

```
for ($i=0; $i<count($tab) ; $i++){
    if ($tab[$i]=="a") {echo $tab[$i], "<br />"; }}
```

- Avec une boucle while

```
$i=0;
while ($tab[$i]){
    if ($tab[$i][0]=="a") {echo $tab[$i], "<br /> "; }}
```

- Avec La boucle foreach

```
$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi");
$i = 0;
foreach($jour as $JJ) { echo "La cellule n° ". $i . " : " . $JJ . "<br>"; $i++; }
```

## Syntaxe de base : Les tableaux(6)

### ■ Lecture des éléments d'un tableau

- Parcours d'un tableau associatif

➤Réalisable en ajoutant avant la variable \$valeur, la clé associée

```
$tableau = array(clé1 => val1, clé2 => val2, cléN => valN);
foreach($tableau as $clé => $valeur) {
    echo "Valeur ($clé): $valeur<br>"; }
```

```
$jour = array("Dimanche" => 7, "Lundi" => 1, "Mardi" => 2, "Mercredi" => 3, "Jeudi" => 4, "Vendredi" => 5, "Samedi" => 6);
foreach($jour as $sJJ => $nJJ) {
    echo "Le jour de la semaine n° ". $nJJ . " : " . $sJJ . "<br>"; }
```

```
Valeur (clé1): val1
Valeur (clé2): val2
Valeur (cléN): valN
Le jour de la semaine n° 7 : Dimanche
Le jour de la semaine n° 1 : Lundi
Le jour de la semaine n° 2 : Mardi
Le jour de la semaine n° 3 : Mercredi
Le jour de la semaine n° 4 : Jeudi
Le jour de la semaine n° 5 : Vendredi
Le jour de la semaine n° 6 : Samedi
```

## Syntaxe de base : Les tableaux(7)

### ■ Fonctions de tri

#### • Tri selon les valeurs

➤ La fonction **sort()** effectue un tri sur les valeurs des éléments d'un tableau selon un critère alphanumérique : selon les codes ASCII :

- « a » est après « Z » et « 10 » est avant « 9 »
- Le tableau initial est modifié et non récupérables dans son ordre original
- Pour les tableaux associatifs les clés seront perdues et remplacées par un indice créé après le tri et commençant à 0

➤ La fonction **rsort()** effectue la même action mais en ordre inverse des codes ASCII.

➤ La fonction **asort()** trie également les valeurs selon le critère des codes ASCII, mais en préservant les clés pour les tableaux associatifs

➤ La fonction **arsort()** la même action mais en ordre inverse des codes ASCII

➤ la fonction **natscasesort()** effectue un tri dans l'ordre alphabétique non ASCII (« a » est avant « z » et « 10 » est après « 9 »)

## Syntaxe de base : Les tableaux(8)

### ■ Fonctions de tri

#### • Tri sur les clés

- La fonction **krsort()** trie les clés du tableau selon le critère des codes ASCII, et préserve les associations clé / valeur

➤ La fonction **krsort()** effectue la même action mais en ordre inverse des codes ASCII

```
<?php
$tab2 = array(1622=>"Molière",1920=>"Vian",1802=>"Hugo");
ksort ($tab2);
var_dump($tab2);
echo "<h3 > Tri sur les clés de " . $tab2. "</h3>";
foreach ($tab2 as $cle=>$valeur) {
echo "<b> l'élément a pour clé : " . $cle. " et pour valeur : " . $valeur
."</b> <br />";
}
?>
```

#### Tri sur les clés de Array

l'élément a pour clé : et pour valeur : Molière

l'élément a pour clé : et pour valeur : Hugo

l'élément a pour clé : et pour valeur : Vian

## Syntaxe de base : Les tableaux(9)

### ■ Les fonctions de tableaux

**\$tableau = array\_count\_values(\$variable);**

retourne un tableau comptant le nombre d'occurrences des valeurs d'un tableau.

**\$tableau = array\_diff(\$var\_1,\$var\_2,..., \$var\_N);**

retourne dans un tableau contenant les valeurs différentes entre deux ou plusieurs tableaux.

**\$tableau = array\_intersect(\$var\_1,\$var\_2,..., \$var\_N);**

retourne un tableau contenant les enregistrements communs aux tableaux entrés en argument.

**\$tableau = array\_flip(\$variable);**

intervertit les paires clé/valeur dans un tableau.

**\$tableau = array\_keys(\$variable [, valeur]);**

retourne toutes les clés d'un tableau ou les emplacements d'une valeur dans un tableau.

## Syntaxe de base : Les tableaux(11)

**\$tableau = array\_filter(\$variable,"fonction")**

retourne un tableau contenant les enregistrements filtrés d'un tableau à partir d'une fonction.

```
<?php
function impair($var)
{return ($var % 2 == 1);}
```

```
function pair($var)
{return ($var % 2 == 0);}
```

```
$array1 = array ("a">1, "b">2, "c">3, "d">4, "e">5);
$array2 = array (6, 7, 8, 9, 10, 11, 12);
echo "Impairs :\n";
print_r(array_filter($array1, "impair"));
echo "Pairs :\n";
print_r(array_filter($array2, "pair"));
?>
```

Impairs : Array ( [a] => 1 [c] => 3 [e] => 5 ) Pairs : Array ( [0] => 6 [2] => 8 [4] => 10 [6] => 12 )



## Syntaxe de base : Les tableaux(1)

### ■ Les fonctions de tableaux

`$tableau = array_map($var_1 [, $var_2, ..., $var_N], 'fonction');`  
applique une fonction à un ou plusieurs tableaux.

`$tableau = array_merge($var_1, $var_2, ..., $var_N);`  
enchaine des tableaux entrés en argument afin d'en retourner un unique.

`$tableau = array_merge_recursive($var_1, $var_2, ..., $var_N);`  
enchaine des tableaux en conservant l'ordre des éléments dans le tableau résultant. Dans le cas de clés communes, les valeurs sont placées dans un tableau.

`true | false = array_multisort($var, critère1, critère2 [, ..., $var_N, critère1, critère2]);`  
trie un ou plusieurs tableaux selon un ordre croissant ou décroissant (SORT\_ASC ou SORT\_DESC) et selon une comparaison alphabétique, numérique ou de chaîne de caractères (SORT\_REGULAR, SORT\_NUMERIC ou SORT\_STRING).

`$tableau = array_pad($variable, taille, valeur);`  
recopie tout un tableau en ajustant sa taille à l'argument correspondant et en bourrant d'une valeur spécifiée les éléments vides.

## Syntaxe de base : Les classes et les objets(1)

### ■ Création d'une classe et d'un objet

#### ● Une classe est composée de deux parties:

- Les attributs: il s'agit des données représentant l'état de l'objet
- Les méthodes: il s'agit des opérations applicables aux objets

```
<?php
class client {
    var $nom; var $ville; var $naiss ;
    function age() {
        $jour = getdate(); $an=$jour["year"]; $age = $an - $this->naiss;
        echo "il a $age ans cette année <br />" ;}}
//création d'un objet
$client1 = new client();
//affectation des propriétés de l'objet
$client1-> nom = "Dupont" ; $client1-> naiss = "1961" ; $client1->ville =
"Angers" ;
//utilisation des propriétés
echo "le nom du client1 est ", $client1->nom, "<br />" ;
echo "la ville du client1 est ", $client1-> ville, "<br />" ;
echo "le client1 est né en ", $client1->naiss, "<br />" ;
//appel de la méthode age()
$client1->age() ;
?>
```

## Syntaxe de base : Les classes et les objets(2)

### ■ Manipulation des classes et des objets

- Instanciation de la classe
  - `$Nom_de_l_objet = new Nom_de_la_classe;`
- Accéder aux propriétés d'un objet
  - `$Nom_de_l_objet->Nom_de_la_donnee_membre = Valeur;`
- Accéder aux méthodes d'un objet
  - `$Nom_de_l_objet->Nom_de_la_fonction_membre(parametre1,parametre2,...);`
- La variable `$this`
  - `$this->age = $Age;`

## Syntaxe de base : Les classes et les objets(3)

### ■ L'héritage

- Instruction extends : `class nouvelle_classe extends super_classe`
- La nouvelle classe hérite des attributs et des méthodes appartenant à la super-classe tout en définissant ses propres fonctions et variables.
- Le langage PHP ne supporte pas l'héritage multiple

#### ● Le constructeur

- Une fonction qui est appelée automatiquement par la classe lors de son instanciation avec l'opérateur `new`
- Doit posséder un nom identique à celle de la classe
- Une fonction constructeur ne peut être définie que dans sa propre classe
- Lorsqu'une classe héritant d'une autre est instanciée et si aucun constructeur n'est défini dans cette classe, alors la fonction constructeur sollicitée sera celle de la super-classe

## Syntaxe de base : Les classes et les objets(4)

### ■ L'opérateur ::

➤faire référence à une fonction définie dans une super-classe à partir d'une classe héritant de cette dernière

```
class nouvelle_classe extends super_classe
{function fonction()
{echo "Blocs d'instructions de la fonction fonction() .
" dans la nouvelle-classe.";
    super_classe::fonction(); }
}
```

## Syntaxe de base : Les classes et les objets(5)

### ■ L'opérateur parent

➤faire référence à des variables ou des fonctions présentes dans la super-classe à partir d'une autre classe héritant de cette dernière

```
class nouvelle_classe extends super_classe
{ function fonction() {
    echo "Blocs d'instructions de la fonction fonction()"
    . " dans la nouvelle-classe.";
    // se réfère à la fonction fonction() de la super_classe
    parent::fonction(); }
}
```

## Syntaxe de base : Les classes et les objets(6)

### ■ Sauvegarde des objets

➤La sauvegarde et la relecture des objets s'effectuent respectivement par **serialize** et **unserialize**

➤**serialize** permet de transformer un objet en une chaîne de caractères pouvant être facilement transmise à une autre page lors d'une session

➤**unserialize** permet de reconstituer l'objet à partir de la chaîne de caractères précitée

## Syntaxe de base : Les classes et les objets(7)

### ■ Sauvegarde des objets : exemple

```
<?php
// Page de définition de la classe trigo_inc.php
class trigonometrie
{
    var $AB;
    var $BC;
    var $AC;
    function hypotenuse()
    {
        $resultat = sqrt(pow($this->BC, 2) + pow($this->AC, 2));
        return number_format($resultat, 2, ',', ' ');
    }
}
?>
```

## Syntaxe de base : Les classes et les objets(8)

### ■ Sauvegarde des objets : exemple suite

```
<?php // Première page : saisie.php
include "trigo_inc.php"; // inclusion de la définition de classe

$trigo = new trigonometrie; // crée une instance de l'objet

$objet_chaine = serialize($trigo); // sérialise l'objet

$fichier = fopen("fic", "w"); // ouvre un fichier en écriture seule

fputs($fichier, $objet_chaine); // écrit l'objet linéarisé dans le fichier

fclose($fichier); // ferme le fichier

?>
```

## Syntaxe de base : Les classes et les objets(9)

### ■ Sauvegarde des objets : exemple suite

```
<form action="resultat.php" method="post">
  <table border="0">
    <tr>
      <th colspan="2">
        <h3>Calcul de l'hypothénuse d'un triangle rectangle</h3>
      </th></tr>
    <tr>
      <td><u>longueur :</u></td>
      <td><input type="text" name="longueur" size="10" maxlength="10">
      </td></tr>
    <tr>
      <td><u>hauteur :</u></td>
      <td><input type="text" name="hauteur" size="10" maxlength="10">
      </td></tr>
    <tr>
      <th colspan="2"><input type="submit" value="Calculer"></th></tr>
  </table>
</form>
```

## Syntaxe de base : Les classes et les objets(10)

```
<?php // Seconde page : resultat.php
include "trigo_inc.php"; // inclusion de la définition de classe
/* regroupe tous les éléments du tableau retourné par la fonction file dans une
chaîne */
$objet_chaine = implode("", file("fic"));
$trigo = unserialize($objet_chaine); // désérialise l'objet
// appelle deux propriétés et une méthode de l'objet
$trigo->BC = $hauteur;
$trigo->AC = $longueur;

?>
<table border="0">
  <tr><th><h3>Calcul de l'hypothénuse d'un triangle rectangle</h3></th></tr>
  <tr><td>hauteur (BC)</td><td>=</td>
    <td><?php echo $trigo->BC ?></td></tr>
  <tr><td>longueur (AC)</td>
    <td>=</td>
    <td><?php echo $trigo->AC ?></td></tr>
  <tr><td>hypothénuse (AB)</td>
    <td>=</td>
    <td><?php echo $trigo->hypothénuse() ?>
  </td></tr>
</table>
```

## Syntaxe de base : Les classes et les objets(11)

### ■ Les fonctions \_\_sleep et \_\_wakeup

➤ Les fonctions \_\_sleep et \_\_wakeup sont appelées resp. par les commandes serialize et unserialize afin de traiter l'objet ou la chaîne de caractères représentant un objet avant la linéarisation ou délinéarisation

```
class nom_classe{
  function __sleep()
  {Instructions à accomplir avant serialize()...}

  function __wakeup() {
    Instructions à accomplir avant unserialize()...
  }
}
```

## Syntaxe de base : Les classes et les objets(12)

### ■ Manipulation des classes et des objets

#### • Les fonctions `__sleep` et `__wakeup`

➤ La fonction **serialize** recherche la méthode `__sleep` dans une classe afin de la lancer avant le processus de linéarisation.

- Effectuer un traitement préliminaire de l'objet dans le but de terminer proprement toutes les opérations relatives à cet objet,
  - la fermeture des connexions sur des bases de données,
  - suppression des informations superflues ne nécessitant pas de sauvegarde, etc..

➤ La fonction **unserialize** recherche la méthode `__wakeup` dans une classe afin de la lancer avant le processus de délinéarisation

- Accomplir des opérations de reconstruction de l'objet
  - en ajoutant des informations,
  - en réouvrant des connexions vers des bases de données,
  - en initialisant des actions, etc..

## Syntaxe de base : Les classes et les objets(13)

### ■ Manipulation des classes et des objets

#### • Les informations méta sur les classes et les objets

➤ <code>Get_class()</code>	détermination de la classe d'un objet
➤ <code>Get_parent_class()</code>	détermination des super-classes d'un objet
➤ <code>Method_exists()</code>	détermination de la présence d'une méthode dans un objet
➤ <code>Class_exists()</code>	Détermination de la présence d'une définition de classe
➤ <code>Is_subclass_of()</code>	Vérifie si une classe est une sous classe d'une autre
➤ <code>Get_class_methods()</code>	Retourne les méthodes d'une classe dans un tableau
➤ <code>Get_declared_classes()</code>	Retourne les classes déclarées dans un tableau
➤ <code>Get_class_vars()</code>	Retourne les variables de classe dans un tableau
➤ <code>Get_object_vars()</code>	Retourne les variables d'un objet dans un tableau

## Syntaxe de base : Les classes et les objets(14)

### ■ Les objets PHP sont des tableaux associatifs

➤ Les noms des variables sont conçus comme des mots-clés

➤ Les valeurs des variables comme les éléments d'un tableau associatif

```
<?
Class ClasseTest {
    var $col = "#0000E0" ;
    var $txt= "Salut PHP" ;
    var $ft = "Arial" ;
    function ClasseTest() {
        echo "<FONT FACE=\ " COLOR=\"\$this->col\" >
        \$this->txt</FONT><br>";
    }
}

$obj = new ClasseTest;
reset($obj);
foreach ($obj as $key=>$elem){
    echo "$key=>$elem<br>" ;
} ?>
```

## La gestion des fichiers avec PHP (1)

### ■ Principe

- PHP prend en charge l'accès au système de fichiers du système d'exploitation du serveur
- Les opérations sur les fichiers concernent la création, l'ouverture, la suppression, la copie, la lecture et l'écriture de fichiers
- Les possibilités d'accès au système de fichiers du serveur sont réglementées par les différents droits d'accès accordés au propriétaire, à son groupe et aux autres utilisateurs
- La communication entre le script PHP et le fichier est repérée par une variable, indiquant l'état du fichier et qui est passée en paramètre aux fonctions spécialisées pour le manipuler

## La gestion des fichiers avec PHP (2)

### ■ Ouverture de fichiers

- La fonction **fopen()** permet d'ouvrir un fichier, que ce soit pour le lire, le créer ou y écrire

**entier fopen(chaine nom du fichier, chaine mode);**

**-mode :** indique le type d'opération qu'il sera possible d'effectuer sur le fichier après ouverture. Il s'agit d'une lettre (en réalité une chaîne de caractères) indiquant l'opération possible:

- r** (comme read) indique une ouverture en lecture seulement
- w** (comme write) indique une ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas)
- a** (comme append) indique une ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)

-lorsque le mode est suivie du caractère **+** celui-ci peut être lu et écrit

-le fait de faire suivre le mode par la lettre **b** entre crochets indique que le fichier est traité de façon binaire.

## La gestion des fichiers avec PHP (3)

### ■ Ouverture de fichiers

Mode	Description
r	ouverture en lecture seulement
w	ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas)
a	ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)
r+	ouverture en lecture et écriture
w+	ouverture en lecture et écriture (la fonction crée le fichier s'il n'existe pas)
a+	ouverture en lecture et écriture avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)

## La gestion des fichiers avec PHP (4)

### ■ Ouverture de fichiers

- Exemple

```
$fp = fopen("fichier.txt","r"); //lecture
```

```
$fp = fopen("fichier.txt","w"); //écriture depuis début du fichier
```

- De plus, la fonction **fopen** permet d'ouvrir des fichiers présents sur le web grâce à leur URL.

➤ Exemple : un script permettant de récupérer le contenu d'une page d'un site web:

```
<?
$fp = fopen("http://www.mondomaine.fr","r"); //lecture du fichier
while (!feof($fp)) { //on parcourt toutes les lignes
    $page .= fgets($fp, 4096); // lecture du contenu de la ligne
}
?>
```

## La gestion des fichiers avec PHP (5)

### ■ Ouverture de fichiers

- Il est généralement utile de tester si l'ouverture de fichier s'est bien déroulée ainsi que d'éventuellement stopper le script PHP si cela n'est pas le cas

```
<?
if (!$fp = fopen("fichier.txt","r")) {
    echo "Echec de l'ouverture du fichier";
    exit;}
else { // votre code;}
?>
```

- Un fichier ouvert avec la fonction **fopen()** doit être fermé, à la fin de son utilisation, par la fonction **fclose()** en lui passant en paramètre l'entier retourné par la fonction **fopen()**

## La gestion des fichiers avec PHP (6)

### ■ Lecture et écriture de fichiers

- Il est possible de lire le contenu d'un fichier et d'y écrire des informations grâce aux fonctions:

➤ **fputs()** (ou l'alias **fwrite()**) permet d'écrire une chaîne de caractères dans le fichier. Elle renvoie 0 en cas d'échec, 1 dans le cas contraire

- booléen **fputs(entier Etat\_du\_fichier, chaîne Sortie);**

➤ **fgets()** permet de récupérer une ligne du fichier. Elle renvoie 0 en cas d'échec, 1 dans le cas contraire

- **fgets(entier Etat\_du\_fichier, entier Longueur);**

- Le paramètre **Longueur** désigne le nombre de caractères maximum que la fonction est sensée récupérer sur la ligne
- Pour récupérer l'intégralité du contenu d'un fichier, il faut insérer la fonction **fgets()** dans une boucle **while**. On utilise la fonction **feof()**, fonction testant la fin du fichier.

## La gestion des fichiers avec PHP (7)

### ■ Lecture et écriture de fichiers

```
<?
if (!$fp = fopen("fichier.txt", "r"))
{echo "Echec de l'ouverture du fichier";}
else { $Fichier="";
while(!feof($fp)) {
// On récupère une ligne
$Ligne = fgets($fp, 255);
// On affiche la ligne
echo $Ligne;
// On stocke l'ensemble des lignes dans une variable
$Fichier .= $Ligne. "<BR>";
}
fclose($fp); // On ferme le fichier
}
?>
```

## La gestion des fichiers avec PHP (8)

### ■ Lecture et écriture de fichiers

- Pour stocker des informations dans le fichier, il faut dans un premier temps ouvrir le fichier en écriture en le créant s'il n'existe pas

➤ Deux choix : le mode 'w' et le mode 'a'.

```
<?
$nom="Jean"; $email="jean@dupont.fr";
$fp = fopen("fichier.txt", "a"); // ouverture du fichier en écriture
fputs($fp, "\n"); // on va à la ligne
fputs($fp, $nom."|".$email); // on écrit le nom et email dans le
fichier
fclose($fp);
?>
```

## La gestion des fichiers avec PHP (9)

### ■ Les tests de fichiers

➤ **is\_dir()** permet de savoir si le fichier dont le nom est passé en paramètre correspond à un répertoire.

- La fonction **is\_dir()** renvoie 1 s'il s'agit d'un répertoire, 0 dans le cas contraire

booléen **is\_dir(chaîne Nom\_du\_fichier);**

➤ **is\_executable()** permet de savoir si le fichier dont le nom est passé en paramètre est exécutable.

- La fonction **is\_executable()** renvoie 1 si le fichier est exécutable, 0 dans le cas contraire

booléen **is\_executable(chaîne Nom\_du\_fichier);**



## La gestion des fichiers avec PHP (10)

### ■ Les tests de fichiers

➤ **is\_link()** permet de savoir si le fichier dont le nom est passé en paramètre correspond à un lien symbolique. La fonction **is\_link()** renvoie 1 s'il s'agit d'un lien symbolique, 0 dans le cas contraire

**booléen is\_link(chaine Nom\_du\_fichier);**

➤ **is\_file()** permet de savoir si le fichier dont le nom est passé en paramètre ne correspond ni à un répertoire, ni à un lien symbolique.

– La fonction **is\_file()** renvoie 1 s'il s'agit d'un fichier, 0 dans le cas contraire

**booléen is\_file(chaine Nom\_du\_fichier);**

## La gestion des fichiers avec PHP (11)

### ■ D'autres façons de lire et écrire

- La fonction **file()** permet de retourner dans un tableau l'intégralité d'un fichier en mettant chacune de ces lignes dans un élément du tableau

➤ **Tableau file(chaine nom\_du\_fichier);**

- Exemple : parcourir l'ensemble du tableau afin d'afficher le fichier

```
<?
$Fichier = "fichier.txt";
if (is_file($Fichier)) {
    if ($TabFich = file($Fichier)) {
        for($i = 0; $i < count($TabFich); $i++)
            echo $TabFich[$i];
        else {echo "Le fichier ne peut être lu...<br>";}
    else {echo "Désolé le fichier n'est pas valide<br> " ;}
?>
```

## La gestion des fichiers avec PHP (12)

### ■ D'autres façons de lire et d'écrire

- La fonction **fpassthru()** permet d'envoyer le contenu d'un fichier dans la fenêtre du navigateur.

**booléen fpassthru(entier etat);**

➤ Elle permet d'envoyer le contenu du fichier à partir de la position courante dans le fichier.

➤ Elle n'ouvre pas automatiquement un fichier. Il faut donc l'utiliser avec **fopen()**.

➤ Il est possible par exemple de lire quelques lignes avec **fgets()**, puis d'envoyer le reste au navigateur.

## La gestion des fichiers avec PHP (13)

**Exemple** : script permettant de parcourir tous les fichiers HTML contenus dans un site à la recherche de MetaTags

```
<?php
function ScanRep($Directory) {
    echo "<b>Parcours</b>: $Directory<br>\n";
    if (is_dir($Directory) && is_readable($Directory)) {
        if($MyDirectory = opendir($Directory)) {
            while($Entry = readdir($MyDirectory)) {
                if (is_dir($Directory."/".$Entry)) {
                    if (($Entry != ".") && ($Entry != "..")) {
                        echo "<b>Repertoire</b>: $Directory/$Entry<br>\n";
                        ScanRep($Directory."/".$Entry);
                    }
                }
                else {echo "<b>Fichier</b>: $Directory/$Entry<br>\n";
                    if (ereg("(.html)|(.htm)", $Entry)) {
                        $meta=get_meta_tags($Directory."/".$Entry);
                        foreach($meta as $cle => $valeur) echo $cle." ".$valeur."<br>\n";
                    }
                }
            }
        }
        closedir($MyDirectory);
    }
    ScanRep(".");
?>
```

## La gestion des fichiers avec PHP (14)

### ■ Le téléchargement de fichier

- Le langage PHP dispose de plusieurs outils facilitant le téléchargement vers le serveur et la gestion des fichiers provenant d'un client
- Un simple formulaire comportant un champ de type file suffit au téléchargement d'un fichier qui subséquemment, devra être traité par un script PHP adapté

```
<form method="POST" action="traitement.php"
    enctype="multipart/form-data">
    <input type="hidden" name="MAX_FILE_SIZE" value="Taille_Octets">
    <input type="file" name="fichier" size="30"><br>
    <input type="submit" name="telechargement" value="telecharger">
</form>
```

## La gestion des fichiers avec PHP(15)

### ■ Le téléchargement de fichier en PHP

- Un champ caché doit être présent dans le formulaire afin de spécifier une taille maximum (MAX\_FILE\_SIZE) pour le fichier à télécharger. Cette taille est par défaut égale à deux mégaoctets.
- En PHP , le tableau associatif global [\\$\\_FILES](#) contient plusieurs informations sur le fichier téléchargé.

```
>$_FILES['fichier']['name'] : fournit le nom d'origine du fichier.
>$_FILES['fichier']['type'] : fournit le type MIME du fichier.
>$_FILES['fichier']['size'] : fournit la taille en octets du fichier.
>$_FILES['fichier']['tmp_name'] : fournit le nom temporaire
                                du fichier.
```

## La gestion des fichiers avec PHP(16)

### ■ Le téléchargement de fichier en PHP

- PHP fait appel au variable globale ou/et au tableau associatif globale \$HTTP\_POST\_VARS à condition que respectivement les options de configuration register\_globals et track\_vars soient activées dans le fichier php.ini.

```
>$fichier : renvoie le nom temporaire du fichier.
>$fichier_name : renvoie le nom d'origine du fichier.
>$fichier_size : renvoie la taille en octets du fichier.
>$fichier_type : renvoie le type MIME du fichier.
>$HTTP_POST_VARS['fichier'] : fournit le nom temporaire du fichier.
>$HTTP_POST_VARS['fichier_name'] : fournit le nom d'origine du fichier.
>$HTTP_POST_VARS['fichier_type'] : fournit le type MIME du fichier.
>$HTTP_POST_VARS['fichier_size'] : fournit la taille en octets du fichier.
```

## La gestion des fichiers avec PHP(17)

### ■ Le téléchargement de fichier

- Par défaut, le fichier envoyé par le client est stocké directement dans le répertoire indiqué par l'option de configuration upload\_tmp\_dir dans le fichier php.ini.  
`upload_tmp_dir = c:\PHP\uploadtemp`
- Plusieurs fonctions spécialisées permettent la validation d'un fichier téléchargé pour son utilisation ultérieure.
  - La fonction `is_uploaded_file` indique si le fichier a bien été téléchargé par la méthode HTTP POST.  
`$booleen=is_uploaded_file($_FILES['fichier']['tmp_name']);`
- La fonction `move_uploaded_file` vérifie si le fichier a été téléchargé par la méthode HTTP POST, puis si c'est le cas le déplace vers l'emplacement spécifié.

## La gestion des fichiers avec PHP(18)

### ■ Le téléchargement de fichier

- Il est possible de télécharger plusieurs fichiers en même temps, en utilisant des crochets à la suite du nom du champ afin d'indiquer que les informations relatives aux fichiers seront stockées dans un tableau.

```
<form action="traitement.php" method="POST" enctype="multipart/form-data">
  <input type="file" name="fichier[]" ><br>
  ...
  <input type="file" name="fichierN[]" > <br>
  <input type="submit" value="Envoyer" name="soumission">
</form>
```

```
for($i = 0; $i < sizeof($_FILES['fichier']['name']); $i++){
  echo "Nom du fichier : ». $_FILES['fichier']['name'][$i];}
```

- Les fichiers téléchargés sont automatiquement effacés du répertoire temporaire au terme du script.
- > il est nécessaire de déplacer les fichiers vers un autre endroit ou de les renommer si ceux-ci doivent être conservés.

## La gestion des fichiers avec PHP(19)

```
<!-- Fichier : formulaire.html -->
<html><body>
  <form method="POST" action="traitement.php" enctype="multipart/form-data">
    <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
    <input type="file" name="fichier" size="30"><br>
    <input type="submit" name="telechargement" value="telecharger">
  </form> </body></html>

<?php /* Fichier : traitement.php */
$repertoire = "f:\\PHP\\uploadtemp";
if (is_uploaded_file($_FILES['fichier']['tmp_name'])) {
  $fichier_temp = $_FILES['fichier']['tmp_name'];
  echo "<h3>Le fichier a été téléchargé avec succès " . "à l'emplacement suivant : <br>"
  . $fichier_temp . "</h3>";
  $nom_fichier = $_FILES['fichier']['name'];
  echo "<h3>Le nom d'origine du fichier est " . $nom_fichier . "</h3>";
  echo "<h3>Le type du fichier est " . $_FILES['fichier']['type'] . "</h3>";
  echo "<h3>La taille du fichier est de " . $_FILES['fichier']['size']
  . " octets.</h3>";
  copy($_FILES['fichier']['tmp_name'], $repertoire . $nom_fichier);
} else {
  echo "<h3 style='color:#FF0000'>ATTENTION, ce fichier peut être à l'origine
  . ' d'une attaque : ' . $_HTTP_POST_FILES['fichier']['name'] . "</h3>";
}>
```

## La gestion des fichiers avec PHP(20)

### ■ La manipulation de fichiers distants

- Il peut être utile de manipuler des fichiers à distance, c'est-à-dire par le biais des protocoles de transferts HTTP ou FTP.
- > PHP autorise l'ouverture d'un fichier par l'intermédiaire d'une adresse URL dans la fonction fopen

```
$id_fichier = fopen("http://www.site.com/index.html", "r");
```

- A partir de ce moment, toutes les informations contenues dans le fichier sont accessibles en lecture seule dans une application PHP

```
$taille = filesize("fichier.html");
echo str_replace("<", "&lt;", fread($id_fichier, $taille));
```

## La gestion des fichiers avec PHP(21)

### ■ La manipulation de fichiers distants

- L'écriture sur un serveur distant est possible, à condition de passer en argument une adresse FTP à la fonction fopen() et que ce fichier soit nouveau.

```
$id_fichier = fopen("ftp://ftp.site.com/new_page.html", "w");
```

- L'accès en écriture directement sur un site, nécessite souvent, la saisie d'un nom d'utilisateur et d'un mot de passe dans l'adresse afin d'éviter toutes intrusions inopportunes

```
ftp://nom_utilisateur:mot_passe@ftp.site.com/nouvelle_page.html
```

- La modification d'un fichier distant n'est pas réalisable par ce moyen

## La gestion des fichiers avec PHP(22)

```
<?php
function recherche_contenu($adresse){
    $id_fichier = fopen($adresse, "r");
    if ($id_fichier)
    {
        $regexp = "<!-- Début contenu -->.*<!-- Fin contenu -->";
        $contenu = fread($id_fichier, filesize($adresse));
        if (mb_ereg($regexp, $contenu, $donnee)){
            echo "<h3><u>Données contenu :</u></h3> "
                . str_replace("<", "&lt;", $donnee[0]);
        }
        else echo "<p>Impossible de le contenu.\n";
    }
    else echo "<p>Impossible d'ouvrir le fichier distant.\n";
    fclose($id_fichier);
}
recherche_contenu("http://www.site.com/page.html");
?>
```

## La gestion des fichiers avec PHP(23)

### ■ Les fonctions de système de fichiers

`$chaine = basename(chemin_fichier);`  
retourne le nom du fichier à partir de l'adresse du fichier spécifiée.

`true | false = chgrp(nom_fichier, groupe_propriétaire);`  
modifie le groupe propriétaire du fichier.

`true | false = chmod(nom_fichier, $mode);`  
modifie le mode exprimé en nombre octal, du fichier.

`true | false = chown(nom_fichier, propriétaire);`  
modifie le groupe propriétaire du fichier.

`clearstatcache();`  
efface la mémoire cache remplie par les fonctions lsat et stat.

`true | false = copy(fichier, nouveau_fichier);`  
copie un fichier vers une nouvelle destination.

## La gestion des fichiers avec PHP(24)

### ■ Les fonctions de système de fichiers

`delete(fichier);`  
efface le fichier.

`$chaine = dirname(chemin);`  
retourne le nom du dossier parent.

`$nombre = disk_free_space(dossier);`  
retourne l'espace disponible sur le disque sur lequel est le dossier.

`$nombre = diskfreespace(dossier);`  
identique à `disk_free_space`.

`$nombre = disk_total_space(dossier);`  
retourne la taille totale d'un dossier.

`true | false = fclose(ID_fichier);`  
ferme un fichier indiqué par un identificateur retourné par `fopen` ou `fsockopen`.

`true | false = feof(ID_fichier);`  
teste la fin du fichier.

## La gestion des fichiers avec PHP(25)

### ■ Les fonctions de dossiers

`nombre = chroot($chaine);`  
définit la chaîne de caractères comme la nouvelle racine

`nombre = chdir($chaine);`  
définit le dossier en cours comme celui précisé par la chaîne de caractères.

`$objet = dir($chaine);`  
crée un objet à partir du dossier spécifié.

`closedir($identificateur_dossier);`  
ferme le dossier à partir d'un identificateur retourné par `opendir`.

`$chaine = getcwd();`  
retourne le nom du dossier en cours.

`$identificateur_dossier = opendir($chaine);`  
ouvre un dossier et retourne un identificateur de l'objet.

`$chaine = readdir($identificateur_dossier);`  
retourne le fichier suivant dans le dossier spécifié par l'entremise de son identificateur.

## Autres Fonctions PHP(1)

### ■ Les dates et les heures

#### • Les fonctions de date et d'heure

`true | false = checkdate(mois, jour, année);`  
vérifie la validité d'une date.

`$chaine = date(format [, nombre]);`  
retourne une chaîne de caractères date/heure selon le format spécifié et représentant la date courante par défaut.

`$tableau = getdate([nombre]);`  
retourne les éléments de date et d'heure dans un tableau associatif.

`$tableau = gettimeofday();`  
retourne l'heure courante dans un tableau associatif.

`$chaine = gmdate(format [, nombre]);`  
retourne une chaîne de caractères date/heure GMT/CUT selon le format spécifié et représentant la date courante par défaut.

`$nombre = gmmktime(heure, minute, seconde, mois, jour, année [, 1/0]);`  
retourne l'instant UNIX d'une date GMT spécifiée et avec éventuellement une heure d'hiver

## Autres Fonctions PHP(2)

### • Les fonctions de date et d'heure

`$chaine = gmstrftime(format [, nombre]);`  
formate une date/heure GMT/CUT en fonction des paramètres locaux définis par setlocale.

`$tableau = localtime([nombre][, tab_associatif]);`  
retourne l'heure locale dans un tableau indicé par défaut ou associatif (1)

`$chaine = microtime();`  
retourne l'instant UNIX courant en secondes et microsecondes (1 janvier 1970 à 0H00)

`$nombre = mktime(heure, minute, seconde, mois, jour, année [, 1/0]);`  
retourne l'instant UNIX d'une date spécifiée et avec éventuellement une heure d'hiver (1)

`$chaine = strftime(format [, instant]);`  
formate une date/heure locale avec les options locales

`$nombre = time();`  
retourne l'instant UNIX courant

## Autres Fonctions PHP(3)

### • Les formats de date et d'heure

a	représente am (matin) ou pm (après-midi).	
A	représente AM (matin) ou PM (après-midi).	
B	représente une heure Internet Swatch.	
d	représente le jour du mois sur deux chiffres allant de 01 à 31.	
D	représente le jour de la semaine en trois lettres et en anglais (Sun, ..., Sat).	
F	représente le mois complet en anglais (January, ..., December).	
g	représente une heure au format 12 heures allant de 1 à 12.	
G	représente une heure au format 24 heures allant de 0 à 23.	
h	représente une heure au format 12 heures avec un zéro de complément allant de 00 à 11.	
H	représente une heure au format 24 heures allant de 00 à 23.	
i	représente les minutes allant de 00 à 59.	
I	est égal à 1 si l'heure d'été est activée ou 0 pour l'heure d'hiver.	
j	représente le jour du mois allant de 1 à 31.	
l	représente le jour de la semaine complet et en anglais (Sunday, ..., Saturday).	
L	est égal à 1 si l'année est bissextile, sinon 0.	
m	représente un mois allant de 01 à 12.	
M	représente un mois en trois lettres et en anglais (Jan, ..., Dec).	

## Autres Fonctions PHP(4)

### • Les formats de date et d'heure

n	représente un mois allant de 1 à 12.	
O	représente la différence d'heures avec l'heure de Greenwich (+0100).	
r	représente un format de date conforme au RFC 822 (Mon, 25 Mar 2002 05:08:26 +0100).	
s	représente les secondes allant de 00 à 59.	
S	représente le suffixe ordinal d'un nombre en anglais et sur deux lettres th ou nd.	
t	représente le nombre de jours dans le mois (28, 29, 30 ou 31).	
T	représente le fuseau horaire.	
U	représente les secondes depuis une époque.	
w	représente le jour de la semaine allant de 0 (Dimanche) à 6 (Samedi).	
Y	représente une année sur quatre chiffres (2002).	
y	représente une année sur 2 chiffres (02).	
z	représente le jour de l'année allant de 0 à 365.	
Z	représente le décalage horaire en secondes.	

## PARTIE II : L'interactivité en PHP

### PHP et les formulaires(1)

#### ■ Formulaire HTML

- Retourne des informations saisies par un utilisateur vers une application serveur
- La création d'un formulaire nécessite la connaissance de quelques balises HTML indispensables :
  - Structure : un formulaire commence toujours par la balise `<form>` et se termine par la balise `</form>`
  - Champ de saisie de text en ligne :  
`<input type = "text" name ="nom_du_champ" value="chaîne">`
  - Boutons d'envoi et d'effacement :  
`<input type=" submit " value = "Envoyer">`  
`<input type = "reset" name ="efface" value = "Effacer">`
  - Case à cocher et bouton radio :  
`<input type = "checkbox" name ="case1" value="valeur_case">`  
`<input type = "radio" name ="radio1" value ="valeur_radio">`

### PHP et les formulaires(2)

#### ➢ Liste de sélection avec options à choix unique :

```
<select name ="select" size="1">
<option value = "un"> choix </option>
<option value = "deux"> choix2 </option>
</select>
```

#### ➢ Liste de sélection avec options à choix multiples :

```
<select name ="select" size = "1" multiple>
<option value = "un"> choix1 </option>
<option value = "deux"> choix2 </option>
</select>
```

### PHP et les formulaires(3)

#### ■ Méthodes d'envoi get et post

##### ➢ transmission selon une des deux méthodes d'envoi GET ou POST

-La méthode GET place les informations d'un formulaire directement à la suite de l'adresse URL de la page appelée.

• <http://www.site.com/cible.php?champ=valeur&champ2=valeur>

##### •inconvénients :

- rendre visibles les données dans la barre d'adresse du navigateur.
- De plus, la longueur totale est limitée à 255 caractères, ce qui rend impossible la transmission d'un volume de données important

-La méthode POST regroupe les informations dans l'entête d'une requête HTTP

- Assure une confidentialité efficace des données



## PHP et les formulaires(4)

### ■ Récupération des paramètres en PHP

- Si la directive `register_globals` de `php.ini` est `TRUE`, lors de la soumission, chaque élément de saisie est assimilé à une variable PHP dont le nom est constitué par la valeur de l'attribut `name` et son contenu par la valeur de l'attribut `value`

A DECONSEILLER

- Les tableaux associatifs `$_GET` et `$_POST` contiennent toutes les variables envoyées par un formulaire
- `$_REQUEST` Un tableau associatif qui contient par défaut le contenu des variables `$_GET`, `$_POST` et `$_COOKIE`.

## PHP et les formulaires(5)

```
<form method="GET | POST" action="page_cible.php">
<input type="text" name="Champ_saisie" value="Texte" >
<select name="Liste_Choix" size="3">
<option value="Option_1">Option_1</option>
<option value="Option_2">Option_2</option>
<option value="Option_3">Option_3</option>
</select>
<textarea name="Zone_Texte" cols="30" rows="5"> Texte par défaut </textarea>
  <input type="checkbox" name="Case_Cocher[]" value="Case_1"> Case 1<br>
  <input type="checkbox" name="Case_Cocher[]" value="Case_2"> Case 2<br>
  <input type="checkbox" name="Case_Cocher[]" value="Case_3"> Case 3<br>
  <input type="radio" name="Case_Radio" value="Case radio 1"> radio 1<br>
  <input type="radio" name="Case_Radio" value="Case radio 2"> radio 2<br>
  <input type="radio" name="Case_Radio" value="Case radio 3"> radio 3<br>
  <input type="reset" name="Annulation" value="Annuler">
  <input type="submit" name="Soumission" value="Soumettre">
</form>
```

## PHP et les formulaires(6)

```
<?php
$resultat = $_GET["Champ_saisie"] . "<br>";
$resultat .= $_GET["Liste_Choix"] . "<br>";
$resultat .= $_GET["Zone_Texte"] . "<br>";
for ($i = 0; $i < count($_GET["Case_Cocher"]); $i++)
{
    $resultat .= $_GET["Case_Cocher"][$i] . "<br>";
}
$resultat .= $_GET["Case_Radio"] . "<br>";
echo $resultat;
?>
```

## PHP et les formulaires(7)

### ■ PHP et les formulaires

- La plupart des éléments d'un formulaire n'acceptent qu'une seule et unique valeur, laquelle est affectée à la variable correspondante dans le script de traitement.

`$Champ_Saisie ← "Ceci est une chaîne de caractères.";`

- Pour des cases à cocher et les listes à choix multiples, plusieurs valeurs peuvent être sélectionnées entraînant l'affectation d'un tableau de valeurs aux variables correspondantes.

`$Case_Cocher[0] ← "Case radio 1";`  
`$Case_Cocher[1] ← "Case radio 3";`

## Les cookies (1)

### ■ Principe

- Un cookie est un fichier texte créé par un script et stocké sur l'ordinateur des visiteurs d'un site
- Les cookies permettent de conserver des renseignements utiles sur chaque utilisateur, et de les réutiliser lors de sa prochaine visite
  - Exemple : personnaliser la page d'accueil ou les autres pages du site
    - un message personnel comportant par exemple son nom, la date de sa dernière visite, ou tout autre particularité.
- Les cookies étant stockés sur le poste client, l'identification est immédiate et ne concernent que les renseignements qui le concernent
- Pour des raisons de sécurité, les cookies ne peuvent être lus que par des pages issues du serveur qui les a créés

## Les cookies(2)

### ■ Principe

- Le nombre de cookies qui peuvent être définis sur le même poste client est limité à 20 et la taille de chacun est limitée à 4ko.
- Un navigateur peut stocker un maximum de 300 cookies
- La date d'expiration des cookies est définie de manière explicite par le serveur web chargé de les mettre en place.
- Les cookies disponibles sont importés par PHP sous forme de variables identifiées sous les noms utilisés par ces cookies
- La variable globale du serveur \$\_COOKIES enregistre tous les cookies qui ont été définis

## Les cookies(3)

### ■ Exemple d'application des cookies

- Mémorisation des paniers dans les applications d'e-commerce
- Identification des utilisateurs
- Des pages web individualisées
  - Afficher des menus personnalisés
  - Afficher des pages adaptées aux utilisateurs en fonction de leurs précédents visites

## Les cookies(4)

### ■ Écrire des cookies

- L'écriture de cookies est possible grâce à la fonction `setcookie()`
  - il faut cette fonction dès le début du script avant l'envoi d'aucune autre information de la part du serveur vers le poste client.

```
setcookie("nom_var", "valeur_var", date_expiration, "chemin", "domain", "secure")
```

```
<?php
setcookie ("PremierCookie", "Salut", time() +3600*24*7) ;
-
if (!$PremierCookie) {
    echo "le cookie n'a pas été défini";
} else {
    echo $premierCookie, "<br>";
}
?>
```

## Les cookies(5)

### ■ Écriture de cookies

- **Nom\_var** : nom de la variable qui va stocker l'information sur le poste client et qui sera utilisée pour récupérer cette information dans la page qui lira le cookie.  
➤ C'est la seule indication obligatoire pour un cookie
- **Valeur\_var** : valeur stockée dans la variable. Par exemple une chaîne de caractères ou une variable chaîne, en provenance d'un formulaire
- **Date\_expiration** : la date à laquelle le cookie ne sera plus lisible et sera effacé du poste client  
➤ on utilise en général la date du jour, définie avec la fonction `time()` à laquelle on ajoute la durée de validité désirée  
➤ Si l'attribut n'est pas spécifié, le cookie expire à l'issue de la session

## Les cookies(6)

### ■ Écriture de cookies

- **Chemin** : définit la destination (incluant les sous-répertoire) à laquelle le navigateur doit envoyer le cookie.
- **Domain set** : le nom du domaine à partir duquel peuvent être lus les cookies.  
➤ On peut aussi utiliser la variable d'environnement `$SERVER_NAME` à la place.
- **Secure** : un nombre qui vaut 0 si la connexion n'est pas sécurisée, sinon, il vaut 1 pour une connexion sécurisée

## Les cookies(7)

### ■ Lecture de cookies

- Si la directive `register_globals` de `php.ini` est `TRUE`, un cookie sera automatiquement passé en paramètre au script et sa valeur sera directement accessible dans la variable `$NomDuCookie`.  
A DECONSEILLER
- Les cookies sont accessibles dans le tableau associatif `$_COOKIE`  
➤ Si celui-ci visite une des pages PHP de ce même domaine dont le chemin est inclut dans le paramètre chemin (si le chemin est / le cookie est valide pour toutes les pages de ce site).  
➤ Il faut d'abord vérifier l'existence des variables dont les noms et les valeurs ont été définis lors de la création du cookie.

- Cette vérification s'effectue grâce à la fonction `isset($_COOKIE["nom_var"])` qui renvoie true si la variable `$nom_var` existe et false sinon.

```
<?php
if (isset($_COOKIE["nom_var"])){
echo "<h2> Bonjour isset($_COOKIE["nom_var"])</h2>" ;}
else echo "pas de cookie" ;
?>
```

## Les cookies(8)

### ■ Écriture de plusieurs variables par un cookie

- Utilisation de la fonction `compact()` pour transformer les variables en un tableau
- Convertir le tableau en une chaîne de caractère à l'aide de la fonction `implode()`
- Affecter la chaîne créée à l'attribut « `nom_cookie` »

```
<?
$col="#FF0000" ;
$size=24;
$font="Arial" ;
$text="Je suis le ookie" ;
$arr=compact("col", " size", " font", "text" );
$val=implode(" &", $arr);
Setcookie("la_cookie", $val, time()+600);
```

## Les cookies(9)

### ■ Lecture de plusieurs variables d'un cookie

- Décomposé la chaîne stockée par la cookie et retrouver un tableau en utilisant la fonction `explode()`

```
<?php
echo " <b> voici le contenu de la chaîne cookie : </b><br>";
echo $le_cookie, "<br> <br>";
}
$arr=explode("&", $la_cookie);
echo "<b> ces variables ont été établies à partir de la chaîne
      cookie : </b> <br> <br>";
foreach ($arr as $k=>$elem) {
echo "$k=>$elem <br>";
}
?>
```

## Les cookies(10)

### ■ Supprimer un cookie

- Il suffit de renvoyer le cookie grâce à la fonction `setcookie()` en spécifiant simplement l'argument `NomDuCookie`

```
<?php
setcookie("Visites");
?>
```

- Une autre méthode consiste à envoyer un cookie dont la date d'expiration est passée:

```
<?php
setcookie("Visites","",time()-1 )
?>
```

## Les cookies(11)

### ■ Quelques précisions sur les cookies

- `setcookie()` doit être utilisée avant l'envoi de données HTML vers le navigateur
- Le cookie n'est pas visible avant le prochain chargement de page
- Avec PHP, si plusieurs cookies sont envoyées de suite, les appels seront traités dans l'ordre
- Certains navigateurs ne traitent pas bien certains cas liés aux cookies
  - Microsoft Internet Explorer 4 avec le Service Pack 1 ne traite pas correctement les cookies qui ont le paramètre chemin défini
  - Netscape Communicator 4.05 et Microsoft Internet Explorer 3.x ne traitent pas correctement les cookies qui n'ont pas les paramètres chemin et expiration définis.

## Les cookies(12)

### ■ Exemples d'utilisation de cookies

```
//Un script permettant de savoir si un visiteur est déjà venu sur le site
pendant le mois
setcookie("Visites","Oui",time()+2592000,"/", ".mondomaine.fr",0);

// Envoi d'un cookie qui disparaîtra après la fermeture du navigateur
SetCookie("UserSessionCookie",$login." ".$pass);

// Envoi d'un cookie qui restera présent 24 heures
SetCookie("DejaVisite","1",time()+3600*24,"/", ".mondomaine.fr ",0);

// Envoi d'un cookie qui s'effacera le 1er janvier 2003
SetCookie("An2002","1",mktime(0,0,0,1,1,2003),"/", ".mondomaine.fr ",0);

//script permettant de compter le nombre de visite de la page par le visiteur
<?php
$Visites++;
setcookie("Visites",$Visites,time()+2592000,"/", ".mondomaine.fr ",0);?>
```

## Les sessions(1)

### ■ Principe

- Est un mécanisme permettant de mettre en relation les différentes requêtes du même client sur une période de temps donnée.
- Les sessions permettent de conserver des informations relatives à un utilisateur lors de son parcours sur un site web
- Des données spécifiques à un visiteur pourront être transmises de page en page afin d'adapter personnellement les réponses d'une application PHP
- Chaque visiteur en se connectant à un site reçoit un numéro d'identification dénommé identifiant de session (SID)
- La fonction `session_start()` se charge de générer automatiquement cet identifiant unique de session et de créer un répertoire. Elle doit être placée au début de chaque page afin de démarrer ou de continuer une session.

```
<?php
session_start();
$Session_ID = session_id();
// $Session_ID = 7edf48ca359ee24dbc5b3f6ed2557e90  ?>
```

## Les sessions(2)

### ■ Principe

- Un répertoire est créé sur le serveur à l'emplacement désigné par le fichier de configuration `php.ini`, afin de recueillir les données de la nouvelle session.

```
[Session]
session.save_path= C:\PHP\sessiondata;
Rép session = \sess_7edf48ca359ee24dbc5b3f6ed2557e90
```

- Le fichier `php.ini` peut également préciser un nom de session par l'option `session.name` ou sa durée de vie par `session.gc_maxlifetime`
- La session en cours peut être détruite par la fonction `session_destroy()`. Cette commande supprime toutes les informations relatives à l'utilisateur.

```
session_destroy();
```

## Les sessions(3)

### ■ Le traitement des variables de session

- Les variables de session sont chargées dans une session par l'intermédiaire de la fonction `session_register()`

```
<?php
session_start();
session_register("nom_variable");
...
session_register("nom_variableN");
?>
```

- Une fois la variable enregistrée, elle est accessible à travers le tableau associatif `$_SESSION["nom_variable"]`

## Les sessions(4)

### ■ Le traitement des variables de session

Le transport des informations entre les documents est réalisé par l'entremise

- soit d'un cookie
- soit d'une requête HTTP

➤ Cette dernière solution est la plus fiable puisque les cookies peuvent ne pas être acceptés par le client ou celui-ci pourrait les détruire en cours de session.

➤ Il suffit de concaténer l'identifiant de session à l'adresse URL de la page cible pour que cette dernière puisse accéder aux informations conservées par la session.

```
echo '<a href=" http://www.site.com/doc.php? '. session_name(). ' = ' . session_id(). '> lien</a>'
```

## Les sessions(5)

### ■ Le traitement des variables de session

- Par défaut, PHP tente de passer par les cookies pour sauvegarder l'identifiant de session dans le cas où le client les accepterait. Il est possible d'éviter cela, il suffit de désactiver l'option de configuration `session.use_cookies` dans le fichier `php.ini`.

[Session]  
`session.use_cookies 0;` // désactive la gestion des sessions par cookie

## Les sessions(6)

### Exemple

```
<!-- Fichier : formulaire.html -->
<html><body>
  <form method="post" action="traitement.php">
    <table border="0">
      <tr>
        <td><u>Nom :</u></td>
        <td><input type="text" name="Nom" size="20" value="RIVES"></td></tr>
      <tr>
        <td><u>Prénom :</u></td>
        <td><input type="text" name="Prenom" size="20" value="Jean-Pierre"></td></tr>
      <tr>
        <td><u>eMail :</u></td>
        <td><input type="text" name="cEmail" size="20" value="du@du.com"></td></tr>
      <tr><td></td>
        <td><input type="submit" name="soumettre" value="Envoyer"></td></tr></table>
    </form>
  </body>
</html>
```

## Les sessions(7)

```
<?
session_start();
$nom = $_POST["Nom"];
$prenom = $_POST["Prenom"];
$email = $_POST["cEmail"];
session_register("nom");
session_register("prenom");
session_register("email");
$_SESSION["nom"]=$nom;
$_SESSION["prenom"]=$prenom;
$_SESSION["email"]=$email;
header("Location: session.php?" . "=". session_id());
?>
```

## Les sessions(8)

```
<?
session_start();
?>
<html><body><?
echo("<u>Identifiant de session :</u> <b>"
    . session_id() . "</b><br>");
echo("<u>Nom de la session :</u> <b>"
    . session_name() . "</b><br><br>");
echo("<u>Nom :</u> <b>" . $_SESSION["nom"] . "</b><br>");
echo("<u>Prénom :</u> <b>" . $_SESSION["prenom"] . "</b><br>");
echo("<u>eMail :</u> <b>" . $_SESSION["email"] . "</b><br>");
//session_destroy();
?>
</body>
</html>
```



## Les sessions (9)

### ■ Les fonctions de sessions

`session_start()` -- Initialise les données de session  
`session_id()` -- Affecte et/ou retourne l'identifiant de session courante  
`session_name()` -- Affecte et/ou retourne le nom de la session courante  
`session_register()` -- Enregistre une variable dans la session courante  
`session_destroy()` -- Détruit toutes les données enregistrées d'une session  
`session_is_registered()` -- Indique si une variable a été enregistrée dans la session ou pas  
`session_unregister()` -- Supprime une variable dans la session courante  
`session_unset()` -- Détruit toutes les variables de session  
`session_cache_expire()` -- Retourne la date d'expiration du cache de la session  
`session_save_path()` -- Affecte et/ou retourne le chemin de sauvegarde de la session courante  
`session_decode()` -- Décode les données de session à partir d'une chaîne  
`session_encode()` -- Encode les données de session dans une chaîne

## Les sessions (10)

### ■ Quelles sont les erreurs possibles ?

#### ● Répertoire de session inaccessible

Warning: open(/tmp/sess\_3c80883ca4e755aa72803b05bce40c12, O\_RDWR)

failed: m (2) in c:\phpdev\www\bp\header.php on line 2

Le répertoire de sauvegarde est défini dans le `php.ini`: `session.save_path = /tmp`

Il faut donc

- Créer un répertoire
- Lui donner les droits d'écriture pour tous
- En spécifier le chemin dans le `php.ini`

#### ● PHP n'est pas autorisé à utiliser les sessions

Il faut s'assurer que le PHP est bien autorisé à créer des sessions. C'est juste un paramètre à activer. Faire un `phpinfo()` pour voir ces paramètres. La commande `phpinfo()` se contente d'afficher dans le navigateur le contenu du fichier de configuration `php.ini`.

## Les sessions (11)

### ■ Quelles sont les erreurs possibles ?

#### ● Avoir déjà écrit dans la page

Warning: Cannot send session cookie - headers already sent by (output started at /home/SiteWeb/SiteAnalyse/index.php:3) in /home/SiteWeb/SiteAnalyse/index.php on line 6

Cette erreur survient lorsqu'on tente d'ouvrir une session après avoir déjà écrit dans le document, ce qui interdit.

Ce qu'il ne faut pas faire :

```
<html>
<body>
<?php session_start();
...
ceci non plus :
<?php echo "<html>";
...
session_start();
```

## Les en-têtes HTTP (1)

### ■ Principe

- Les entêtes sont des informations envoyées lors de chaque échange par le protocole HTTP entre un navigateur et un serveur
  - Informations sur les données à envoyer dans le cas d'une requête
- Permettent aussi d'effectuer des actions sur le navigateur comme le transfert de cookies ou bien une redirection vers une autre page
  - Ce 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
  - 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

## Les en-têtes HTTP (2)

### ■ Principe

- PHP fournit une fonction permettant d'envoyer des en-tête HTTP manuellement du serveur au navigateur

➤ booléen header(chaine en-tête HTTP)

- La fonction header() doit être utilisée avant tout envoi de données HTML au navigateur

### • Exemples

➤ Rediriger le navigateur vers une nouvelle page:

```
<?
header("location: http://www.monsite.fr/"); ?>
```

➤ Pour envoyer au navigateur une image créée à la volée

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

## Les en-têtes HTTP (2)

### ■ Principe

- PHP fournit une fonction permettant d'envoyer des en-tête HTTP manuellement du serveur au navigateur

➤ booléen header(chaine en-tête HTTP)

- La fonction header() doit être utilisée avant tout envoi de données HTML au navigateur

### • Exemples

➤ Rediriger le navigateur vers une nouvelle page:

```
<?
header("location: http://www.monsite.fr/"); ?>
```

➤ Pour envoyer au navigateur une image créée à la volée

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

## Les en-têtes HTTP (3)

### ■ 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

Tableau getallheaders();

➤ Le tableau retourné par la fonction contient les en-têtes indexés par leur nom

➤ Exemple : un script permettant par exemple de récupérer des en-têtes particuliers.

```
<?
$headers = getallheaders();
foreach ($headers as $nom => $contenu) {
    echo "headers[$nom] = $contenu<br />";
}
?>
```

## Les en-têtes HTTP (4)

### ■ Les fonctions HTTP

header(chaine);

envoie une entête HTTP avant toute commande PHP.

true | false = headers\_sent();

vérifie si les entêtes HTTP ont bien été envoyés

setcookie(nom, valeur [, date\_expiration [, chemin [, domaine [, sécurisation]]]]);

envoie un cookie.

## Création d'images à la volée (1)

- PHP ne se limite pas à la génération de pages HTML
- Il peut aussi servir à créer et manipuler d'autres types de documents
- Notamment des images, au travers dans un grand choix de formats, comme GIF , PNG , JPEG , WBMP...
- PHP peut même générer directement des images pour le navigateur, avec la bibliothèque GD .
- GD et PHP auront aussi besoin d'autres bibliothèques, en fonction des formats que vous souhaitez utiliser.
- Vous pouvez utiliser les fonctions PHP pour obtenir les tailles des images aux formats JPEG , GIF , PNG , SWF , TIFF et JPEG2000 .

## Création d'image à la volée (2)

- PHP doit définir, dans l'entête HTTP, le type du document envoyé au navigateur

fichier rotation\_test.php

```
<?php
```

```
header("Content-type: image/png");
```

```
$im = imagecreatefrompng("test.png");
```

```
$im = imagerotate($im, 90, 0);
```

```
imagepng($im);
```

```
?>
```

fichier test.htm

```
<html><body><img src='rotation_test.php'></body></html>
```

## Création d'image à la volée (3)

- Chargement d'images depuis des fichiers existants
  - imagecreatefromgif ( string filename )
  - imagecreatefromjpeg ( string filename )
  - imagecreatefrompng ( string filename )
  - imagecreatefromwbmp ( string filename )
  - imagecreatefromstring ( string filename )
    - le type est automatiquement détecté
- Création d'une image vierge
  - imagecreate ( int x\_size , int y\_size )
  - imagecreatetruecolor ( int x\_size , int y\_size )

## Création d'image à la volée (4)

- Redimensionnement

```
int imagecopyresized ( resource dst_image , resource src_image , int dst_x , int dst_y ,
    int src_x , int src_y , int dst_w , int dst_h , int src_w , int src_h )
```

imagecopyresized copie une partie rectangulaire d'une image dans une autre image.

Si les dimensions de la source et de la destination ne sont pas égales, un étirement adéquat est effectué pour faire correspondre

```
bool imagecopyresampled ( resource dst_image , resource src_image , int dst_x , int
    dst_y , int src_x , int src_y , int dst_w , int dst_h , int src_w , int src_h )
```

copie une zone rectangulaire de l'image src\_im vers l'image dst\_im . Durant la copie, la zone est rééchantillonnée de manière à conserver la clarté de l'image .

Cette fonction retourne TRUE en cas de succès, FALSE en cas d'échec.

## Création d'image à la volée (4)

- Dessiner sur une image
- imagechar ( resource image , int font , int x , int y , string c , int color )
- imagestring ( resource image , int font , int x , int y , string c , int color )
- imagearc ( resource image , int cx , int cy , int w , int h , int s , int e , int color )

## Mail (1)

- La fonction mail envoie un message électronique.
- Syntaxe :
- ```
mail($to, $subject, $message[, $headers, $params]);
```

Exemple :

```
$message = "Bonjour, ceci est mon message.";
$objet = "Bonjour"
mail("info@aricia.fr", $objet, $message);
```

Cette fonction ne marche que si un programme de messagerie électronique (appelé « mailer ») est préalablement installé sur le serveur.

## Mail (2)

- Exemple plus complet :

```
<?php
$to = "Tony <tony@tony.com>";
$to = "Peter <peter@peter.net>";
$subject = "Notre rendez-vous";
$message = "Je vous propose le samedi 15 juin \n";
$message .= "\n"; // Délimiteur de signature
$message .= "Hugo";
$headers = "From: Hugo <hugo@hugo.com>\n";
$headers .= "Content-Type: text/html; charset=iso-8859-1\n";
$headers .= "Cc: bruno@aricia.fr\n";
mail($to, $subject, $message, $headers);
?>
```

## Principes de sécurité (1)

- veiller à limiter au strict nécessaire les droits et permissions de l'utilisateur : il vaut mieux l'entendre se plaindre de son manque de liberté plutôt que de constater les dégâts causés par une trop grande libéralité (principe universel)
- ne jamais faire confiance aux données transmises par l'utilisateur (encore moins à l'internaute !)
  - SQL Injection
  - Javascript injection
- toujours tester l'existence / la validité d'un fichier / code à inclure : et s'il n'était pas celui que vous croyiez ?
- Regarder régulièrement les fichiers journaux (logs)
- Mettre des "alarmes" dans votre code, aux endroits stratégiques (authentification), envoyées par mail ou stockées dans une base de données

## Principes de sécurité (2)

- un contrôle des données côté client est bien pratique, mais illusoire : quoi de plus simple que de désactiver Javascript ? Seul un contrôle côté serveur peut prétendre être efficace.
- préférer, quand cela est possible, la méthode POST à la méthode GET : les variables dans l'url, ça fait mauvais genre
- ne pas confondre complexité et sécurité : si votre système de sécurité vous échappe, considérez que vous n'êtes pas en sécurité. De toute façon, vous n'êtes jamais en sécurité.
- s'efforcer de tenir à jour système serveur et interpréteur PHP avec les dernières versions stables : quand la catastrophe aura eu lieu, ce sera ça de moins à vous reprocher...

## Principes de sécurité (3)

### ■ Masquer PHP

Configurer le serveur web pour qu'il utilise plusieurs extensions de fichiers avec PHP

Vous pouvez utiliser des informations déroutantes comme ceci :

- **Masquer PHP avec un autre langage**

```
# Faire que le code PHP ressemble à un autre langage
AddType application/x-httpd-php .asp .py .pl
```

- **Masquer PHP avec des types inconnus**

```
# Faire que le code PHP ressemble à un autre langage qui n'existe pas
AddType application/x-httpd-php .bop .foo .l33t
```

- **cachez-le sous forme de html.**

Cela a un léger impact négatif sur les performances générales, car tous les fichiers HTML seront aussi analysés et traités par le moteur PHP :

Utiliser le type html pour les extensions PHP

```
# Faire que le code PHP ressemble à du html
AddType application/x-httpd-php .htm .html
```

## Interfaçage avec une base de données(1)

### ■ Principe

- PHP propose de nombreux outils permettant de travailler avec la plupart des SGBDR
  - Oracle, Sybase, Microsoft SQL Server, PostgreSQL ou encore MySQL
- Lorsqu'une base de données n'est pas directement supportée par Php, il est possible d'utiliser un driver ODBC (pilote standard) pour communiquer avec les bases de données
- Php fournit un grand choix de fonctions permettant de manipuler les bases de données.
  - Quatre fonctions sont essentielles:
    - La fonction de connexion au serveur
    - La fonction de choix de la base de données
    - La fonction de requête
    - La fonction de déconnexion

## Interfaçage avec une base de données(2)

### ■ Principe

- Avec le SGBD MySQL, les fonctions de manipulation sont :
  - `mysqli_connect`
  - `mysqli_select_db`
  - `mysqli_query`
  - `mysqli_close`

## Interfaçage avec une base de données(3)

### ■ La connexion à un SGBDR

- Déclarer les variables qui vont permettre la connexion à la base de données

➤ Ce sont les paramètres (optionnels généralement) des fonctions de connexion à la base. Ces variables sont:

- \$user : Le nom d'utilisateur
- \$passwd : Le mot de passe
- \$host : L'hôte (ordinateur sur lequel le SGBD est installé)
  - à défaut le nom d'hôte est localhost
- \$bdd : Le nom de la base de données

## Interfaçage avec une base de données(4)

### ■ La connexion à un SGBDR

- Déclarer les variables qui vont permettre la connexion à la base de données

➤ Il est utile de créer un script PHP contenant les variables de connexion sous forme de variables et lier ce script aux autres scripts en utilisant l'instruction include().

```
<? // connexion_data.php
/* Accès au serveur de bases de données MySQL */
$MySQL_Hot="localhost";
$MySQL_User="admin";
$MySQL_Passw="admin";
?>
```

## Interfaçage avec une base de données(5)

### ■ La connexion à un SGBDR

- La connexion à (un serveur) un système de gestion de base de données s'effectue par l'entremise des fonctions spécialisées

➤ `mysqli_connect("nom_serveur","user","mot_passe","dbname");`

–retourne une valeur de type Integer servant d'identificateur de connexion lorsque la connexion a été établie  
–retourne False si une erreur se produit lors de l'établissement de la connexion

➤ `pg_connect("dbname=nom_base,host=nom_serveur port=num_port" "user=nom_utilisateur password=mot_passe");`

Pour une connexion à une base de données PostgreSQL

## Interfaçage avec une base de données(6)

### ■ La connexion à un SGBDR

- Il existe deux façons de se connecter à une base de données

➤ Les connexions non-persistantes (base\_connect)

➤ Les connexions persistantes (base\_pconnect).

- Les deux types de connexions sont parfaitement identiques au niveau des fonctionnalités qu'elles apportent

- Néanmoins, les connexions persistantes ne se referment pas automatiquement à la fin du script

➤ PHP s'assure qu'il n'existe pas un processus semblable, déjà ouvert avec les noms de serveur et d'utilisateur ainsi que le mot de passe. Si tel est le cas, ce processus est réutilisé sinon un nouveau est ouvert

➤ le principal avantage des connexions persistantes est leur réutilisabilité



## Interfaçage avec une base de données(7)

### ■ La connexion à un SGBDR

- La déconnexion des bases de données s'effectue par l'intermédiaire des fonctions de fermeture

```
>mysqli_close($id_connexion);
>mssql_close($id_connexion);
```

- Plusieurs fonctions PHP permettent de retourner des informations à propos de la connexion en cours

```
>$chaîne_numéro_version = mysqli_get_client_info();
>$type_connexion = mysqli_get_host_info($id_connexion);
>$protocole_connexion = mysqli_get_proto_info($id_connexion);
>$chaîne_version_serveur = mysqli_get_server_info($id_connexion);
>$nom_hôte = pg_host($id_connexion);
>$option_connexion = pg_options($id_connexion);
>$num_port = pg_port($id_connexion);
>$encodage = pg_client_encoding($id_connexion);
```

## Interfaçage avec une base de données(8)

```
<html> body> <!-- Fichier : formulaire.php -->
<form action="traitement.php" method="POST">
<table>
<tr> <td>Nom </td>
      <td> <input type="text" size="20" name="nom_utilisateur"></td> </tr>
<tr> <td>Mot de passe</td>
      <td> <input type="password" size="20" name="mot_de_passe"></td> </tr>
<tr> <td>Serveur hôte</td>
      <td> <input type="text" size="20" name="nom_serveur" value="<?php echo
$SERVER_NAME ?>"></td> </tr>
<tr> <td>Numéro de port</td>
      <td> <input type="text" size="20" name="num_port" value="<?php echo
$SERVER_PORT ?>"></td> </tr>
<tr> <td>Nom SGBDR</td>
      <td> <input type="text" size="20" name="nom_sgbdr"></td> </tr>
<tr> <td>Autre paramètre </td>
      <td> <input type="text" size="20" name="autre_param"></td> </tr>
<tr> <td>
      <td> <input type="submit" name="Soumission" value="Soumettre"> </td> </tr>
</table> </form> </body></html>
```

## Interfaçage avec une base de données(9)

```
<?php
// Fichier : traitement.php

function connexion($sgbdr, $hôte, $port, $utilisateur, $mot_passe,$param_sup){
if ($type == "") {
    echo ("Aucun SGBDR n'a été spécifié !");
    return false;
}
else {
    switch ($type) {
        case "MySQL" : {
            if ($port != "") $hôte .= ":" . $port;
            $id_connexion = mysqli_connect($hôte, $utilisateur, $mot_passe, bdd);
        }
    }
}
```

## Interfaçage avec une base de données(10)

```
case "ODBC" : {
    if ($param_sup == "")
        $id_connexion = odbc_connect($hôte,$utilisateur,$mot_passe,
        $param_sup);
    else $id_connexion = odbc_connect($hôte,$utilisateur,$mot_passe,
        $param_sup);
}
case "Oracle" : { $id_connexion = oci_logon($utilisateur,$mot_passe); }
case "PostgreSQL" : { $chaîne_connexion = "host=" . $hôte . " ";
    if ($port != "") $chaîne_connexion .= "port=" . $port . " ";
    $chaîne_connexion .= "user=" . $utilisateur . " password=" . $mot_passe;
    if ($param_sup != "") $chaîne_connexion .= " " . $param_sup;
    $id_connexion = pg_connect($chaîne_connexion); }
case "Sybase" : {
    if ($param_sup == "")
        $id_connexion = sybase_connect($hôte,$utilisateur,$mot_passe);
    else $id_connexion = sybase_connect($hôte,$utilisateur,$mot_passe,$param_sup); }
default : { echo ("Le SGBDR " . $sgbdr . " n'est pas géré.");
    return false; } }
return true; } }
connexion($nom_sgbdr,$nom_serveur,$num_port, $nom_utilisateur,$mot_de_passe,
    $autre_param);
?>
```

## Interfaçage avec une base de données(11)

### ■ L'accès aux bases de données

- Suite à la connexion à un SGBDR, il faut soit sélectionner la base de données si elle existe déjà, soit la créer.

- La sélection de bases de données s'effectue par des fonctions adaptées :

```
>mysqli_select_db($nom_base_donnee,$id_connexion);
>mysqli_select_db($nom_base_donnee,$id_connexion);
>sybase_select_db($nom_base_donnee,$id_connexion);
>pg_connect("dbname=nom_base"."host=nom_serveur"."port=num_port".
"user=nom_utilisateur"."password=mot_passe");
```

- La création des bases de données peut être réalisée par des fonctions PHP dévolues à cette tâche

```
>mysqli_create_db($nom_base_donnee,$id_connexion);
>mysqli_create_db ($nom_base_donnee, $id_connexion);
```

## Interfaçage avec une base de données(12)

### ■ L'accès aux bases de données

- Création d'une base de données

>la fonction `create_db()` attend deux paramètres

- Une variable de type String portant le nom de la base de données à créer
- L'identificateur de connexion de la connexion courante

```
function creatdb($db, $lkid) {
if (! $res=mysqli_create_db($db, $lkid) )
echo mysqli_error($lkid);
exit;
}
return $res;
}
?>
```

## Interfaçage avec une base de données(13)

### ■ L'accès aux bases de données

- Si la création d'une base de données n'est pas possible à l'aide de fonctions, il est possible de créer une base à l'aide d'une requête SQL.

```
$requete = "CREATE DATABASE nom_base_donnee";
$id_requete = ociparse($requete, $id_connexion);
ociexecute($id_requete);
```

- La suppression des bases de données est permise, de la même façon qu'il est possible de les créer.

```
> msqldrop_db($nom_base_donnee, $id_connexion);
> mysqli_drop_db ($nom_base_donnee, $id_connexion);
```

## Interfaçage avec une base de données(14)

### ■ L'accès aux bases de données

- Certaines fonctions permettent de retourner la liste des bases de données et de leurs tables.

```
$id_connexion = mysqli_connect('localhost','administrateur','md2N500ml','bdd');
$liste_bases = mysqli_list_dbs($id_connexion);
$nb_bases = mysqli_num_rows($liste_bases);
echo "<h3>Liste des bases de données</h3>";
for($i = 0; $i < $nb_bases; $i++){
    $nom_base_donnee = mysqli_db_name($liste_bases, $i);
    $liste_tables = mysqli_list_tables($nom_base_donnee, $id_connexion);
    $nb_tables = mysqli_num_rows($liste_tables);
    echo "<h3>". $nom_base_donnee . "</h3>";
    echo "<h4>Liste des tables :</h4>";
    for($j = 0; $j < $nb_tables; $j++) {
        echo mysqli_tablename($liste_tables, $j) . "<br>";
    }
}
```

## Interfaçage avec une base de données(15)

### ■ Les requêtes SQL

- Les requêtes doivent répondre à la syntaxe SQL en général et éventuellement aux singularités des différents éditeurs de SGBDR.
- Les requêtes SQL permettent d'accomplir une action sur une base de données comme
  - la sélection d'informations,
  - la création de tables,
  - l'ajout, la suppression ou la modification des enregistrements.
- Les requêtes SQL sont envoyées à la base de données définie par un identificateur de connexion

```
$requete = "SELECT * FROM table WHERE champ = \"valeur\"";
$id_resultat = mssql_query($requete, $id_connexion);
$id_resultat1 = mysqli_query($requete, $id_connexion);
```

## Interfaçage avec une base de données(16)

### ■ Les requêtes SQL

```
>$requete = "CREATE TABLE tbl_nom ("
            . "nom_champ1 INTEGER PRIMARY KEY,"
            . "nom_champ2 CHAR(50) UNIQUE,"
            . "nom_champ3 DATETIME)";

>$requete = "INSERT INTO tbl_nom "
            . "(nom_champ1, nom_champ2, nom_champ3) "
            . "VALUES('valeur', 'valeur2', 'valeur3')";

>$requete = "SELECT * FROM tbl_nom "
            . "WHERE nom_champ2 = 'valeur'";

>$requete = "DELETE FROM tbl_nom "
            . "WHERE nom_champ3 < SYSDATE - 7";
```

## Interfaçage avec une base de données(17)

### ■ Traitement des résultats d'une requête

- La variable contenant l'ensemble des enregistrements retournés par une requête n'est pas exploitable telle quelle.
  - On utilise la fonction `mysqli_fetch_row()`, qui découpe les lignes de résultat en colonnes (par exemple Nom, adresse,...) et les affecte à une variable tableau dans l'ordre où elles arrivent.
- Exemple: une table appelée liens contenant le nom et l'URL de sites internet.
  - récupérer l'ensemble des enregistrements et de les afficher dans un tableau

## Interfaçage avec une base de données(18)

```
<html>
<head>
<title>Liens</title>
</head>
<body>
<table border="1" cellpadding="0" cellspacing="0">
<tr>
<th>Nom du site</th>
<th>URL</th>
</tr>
</table>
<?php
// Déclaration des paramètres de connexion
$host = la_machine;
$user = votre_login;
$dbdd = Nom_de_la_base_de_donnees;
$password = Mot_de_passe;
// Connexion au serveur
mysqli_connect($host, $user, $password) or die("erreur de connexion au serveur");
mysqli_select_db($dbdd) or die("erreur de connexion à la base de donnees");
```

## Interfaçage avec une base de données(19)

```
// Creation et envoi de la requete
$query = "SELECT nom,url FROM sites ORDER BY nom";
$result = mysqli_query($query);
// Recuperation des resultats
while($row = mysqli_fetch_row($result)){
    $Nom = $row[0];
    $Url = $row[1];
    echo "<tr>\n
    <td><a href=\"\$Url\">\"$Nom</a></td>\n
    <td>$Url</td>\n
    </tr>\n";
}
// Déconnexion de la base de données
mysqli_close();
?>
</tr>
</table>
</body>
</html>
```

## Interfaçage avec une base de données(20)

### ■ Les propriétés de colonnes

- Certaines instructions permettent de récupérer diverses informations à propos des champs d'une table résultant d'une requête SQL.

➤ Le **nom d'un champ** peut être retourné par l'intermédiaire de certaines fonctions.

```
$nom = mysql_fieldname($id_resultat,$num_champ);
$nom = mysqli_field_name($id_resultat,$num_champ);
$nom = ociColumnName($id_resultat,$num_champ);
$nom = pg_FieldName($id_resultat,$num_champ);
```

➤ Le **type du champ** est obtenu en utilisant des fonctions spécifiques.

```
$type_donnee = mysql_fieldtype($id_resultat,$num_champ);
$type_donnee = mysqli_field_type($id_resultat,$num_champ);
$type_donnee = ociColumnType($id_resultat,$num_champ);
$type_donnee = pg_FieldType($id_resultat,$num_champ);
```

## Interfaçage avec une base de données(21)

### ■ Les propriétés de colonnes

- D'autres fonctions renvoient la longueur d'un champ

```
$longueur = mssql_field_length($id_resultat,$num_champ);
$longueur = mysql_fieldlen($id_resultat,$num_champ);
$longueur = mysqli_field_len($id_resultat,$num_champ);
$longueur = ociColumnSize($id_resultat,$num_champ);
```

- D'autres propriétés peuvent être recueillies par des instructions relatives à certains gestionnaires de bases de données

```
// nom de la table parente de la colonne
$nom_table = mssql_fieldtable($id_resultat,$num_champ);
$nom_table = mysql_tablename($id_resultat,$num_champ);
$nom_table = mysqli_field_table($id_resultat,$num_champ);
$nom_table = mysqli_tablename($id_resultat,$num_champ);

// sémaphore du champ comme NOT NULL, PRIMARY KEY
$semaphore = mysql_fieldflags($id_resultat,$num_champ);
$semaphore = mysqli_field_flags($id_resultat,$num_champ);
```

## Interfaçage avec une base de données(22)

```
<?php
$id_connexion = mysqli_connect("localhost","root","emma");
mysqli_select_db($id_connexion,"utilisateur");
$query = "SELECT * FROM tbl_utilisateur";
$resultat = mysqli_query($query,$id_connexion);
or die ("La requête est invalide : ".$query."<br>");
$nb_champs = mysqli_num_fields($resultat);
$ligne = mysqli_fetch_row($resultat);
$type = array();
$propriete = array();
for ($i = 0; $i < $nb_champs; $i++) {
    $propriete[$i]['nom'] = mysqli_field_name($resultat,$i);
    $propriete[$i]['type'] = mysqli_field_type($resultat,$i);
    $propriete[$i]['longueur'] = mysqli_field_len($resultat,$i);
}
for ($i = 0; $i < $nb_champs; $i++) {
    echo "<h3>Colonne n°" . $i . " :</h3>";
    foreach($propriete[$i] as $cle => $valeur){
        echo "<u> . $cle . " :</u> <b> . $valeur . " </b><br>";
    }
}
?>
```

## Interfaçage avec une base de données(23)

### ■ L'exploitation des données

- Le décompte total des enregistrements ou des champs d'une table, peut être obtenu par l'intermédiaire de certaines fonctions

```
$nb_champs = msql_num_fields($id_resultat);
$nb_lignes = msql_num_rows($id_resultat);
```

```
$nb_champs = mysqli_num_fields($id_resultat);
$nb_lignes = mysqli_num_rows($id_resultat);
```

```
$nb_champs = ociNumCols($id_resultat);
$nb_lignes = ociRowCount($id_resultat);
```

```
$nb_champs = pg_NumFields($id_resultat);
$nb_lignes = pg_NumRows($id_resultat);
```

```
$nb_champs = sybase_num_fields($id_resultat);
$nb_lignes = sybase_num_rows($id_resultat);
```

## Interfaçage avec une base de données(24)

### ■ L'exploitation des données

- A partir des enregistrements, chacun des champs devient accessible aisément par des fonctions appropriées. Certaines fonctions sont capables d'extraire directement un champ déterminé

```
$valeur = mssql_result($id_resultat, $num_ligne, $num_col);
$valeur = msql_result($id_resultat, $num_ligne, $num_col);
$valeur = mysqli_result($id_resultat, $num_ligne, $num_col);
$valeur = pg_result($id_resultat, $num_ligne, $num_col);
$valeur = sybase_result($id_resultat, $num_ligne, $num_col);
```

- Afin d'éviter une surcharge de la suite au terme de leur utilisation, mémoire, des instructions PHP permettent de libérer les ressources

```
mssql_free_result($id_resultat);
msql_free_result($id_resultat);
mysqli_free_result($id_resultat);
sybase_free_result($id_resultat);
```

## Interfaçage avec une base de données(25)

### ■ La gestion des erreurs

- Les erreurs générées par la plupart des SGBDR ne sont plus traitées comme des alertes. Elles sont stockées et disponibles à partir d'une fonction spécifique
- Il est possible d'interrompre le script afin d'éviter les erreurs en cascade. Deux méthodes permettent d'effectuer cette opération

➤ Le stockage du résultat de l'exécution de la fonction dans une variable

```
$connect = mysqli_connect($host,$user,$password,$bdd);
```

➤ L'utilisation de la fonction `die()` en cas d'erreur d'exécution. Si la fonction retourne la valeur 0 (c'est-à-dire s'il y a une erreur) la fonction `die()` renvoie un message d'erreur.

```
mysqli_connect($host,$user,$password,$bdd) or die("erreur de connexion au serveur $host");
```

➤ En général, deux fonctions PHP retournent respectivement le **numéro** et le **message de l'erreur en cours**. Certains SGBDR ne gèrent que le message d'erreur tels que PostgreSQL et Sybase.

## Interfaçage avec une base de données(26)

### ■ La gestion des erreurs

```
// message d'erreur généré par Microsoft SQL Server
$message = mssql_get_last_message();

// numéro et message d'erreur due à la dernière action pour MySQL
$num_erreur = mysqli_errno($id_connexion);
$message = mysqli_error($id_connexion);

// message d'erreur généré par mSQL
$message = msql_error();

// message d'erreur pour une connexion ou une requête Oracle 8
$message = ociError($id_connexion | $id_requete);

// message d'erreur généré par PostgreSQL
$message = pg_ErrorMessage($id_connexion);

// message d'erreur généré par Sybase
$message = sybase_get_last_message();
```



## Interfaçage avec une base de données(27)

```
<?php
$id_connexion = mysqli_connect("localhost", "root", "mot");
if (!$id_connexion)
{
    $message = "<h3>Une erreur est survenue :</h3>" . "<b><u>Erreur numéro "
        . mysqli_errno() . " :</u>" . mysqli_error() . "</b>";
    echo $message;
}
else {$reussite = mysqli_select_db("utilisateur");
    if (!$reussite)
    {
        $message = "<h3>Une erreur est survenue :</h3>" . "<b><u>Erreur numéro "
            . mysqli_errno() . " :</u>" . mysqli_error() . "</b>";
        echo $message;
    }
    else {$id_requete = mysqli_query("SELECT date, email, nom "
        . "FROM tbl_utilisateur");
        if (!$id_requete) {
            $message = "<h3>Une erreur est survenue :</h3>" . "<b><u>Erreur numéro "
                . mysqli_errno() . " :</u>" . mysqli_error() . "</b>";
            echo $message;
        }
    }
}
}}?>
```

## Interfaçage avec une base de données(28)

```
<?php
function gestion_erreur($identifiant){
    echo "Une erreur est survenue :<br>"
        . mysqli_errno() . " : " . mysqli_error();
    mysqli_close($identifiant);
    exit;
}
$id_connexion = mysqli_connect("localhost", "root", "mpt");
if (!$id_connexion) gestion_erreur($id_connexion);
$id_liste_bases = mysqli_list_dbs($id_connexion);
if (!$id_liste_bases) gestion_erreur($id_liste_bases);
$nb_bases = mysqli_num_rows($id_liste_bases);
echo '<table width="640" border="1">';
for($i = 0; $i < $nb_bases; $i++){
    $nom_base = mysqli_db_name($id_liste_bases, $i);
    $id_liste_tables = mysqli_list_tables($nom_base, $id_connexion);
    if (!$id_liste_tables) gestion_erreur($id_liste_tables);
    $nb_tables = mysqli_num_rows($id_liste_tables);
    echo '<tr><th valign="top" width="140"><u>' . $nom_base
        . '</u></th><td valign="top" width="500">';
```

## Interfaçage avec une base de données(29)

```
for($j = 0; $j < $nb_tables; $j++){
    $nom_table = mysqli_tablename($id_liste_tables, $j);
    $id_liste_champs = mysqli_list_fields($nom_base, $nom_table, $id_connexion);
    if (!$id_liste_champs) gestion_erreur($id_liste_champs);
    $nb_champs = mysqli_num_fields($id_liste_champs);
    echo '<table width="500" border="1"><tr><th valign="top" width="200">'
        . $nom_table . '</th><td valign="top" width="300">';
    for($k = 0; $k < $nb_champs; $k++){
        $nom_champ = mysqli_field_name($id_liste_champs, $k);
        $type_champ = mysqli_field_type($id_liste_champs, $k);
        $longueur_champ = mysqli_field_len($id_liste_champs, $k);
        $semaphore_champ = mysqli_field_flags($id_liste_champs, $k);
        echo '<u>' . $nom_champ . '</u>('
            . $type_champ . ' '
            . $longueur_champ . ' '
            . $semaphore_champ . ')<br>';
    }
    echo '</td></tr></table>'; echo '</td></tr>'; } echo '</table>';
mysqli_free_result($id_liste_bases);
mysqli_free_result($id_liste_tables);
mysqli_free_result($id_liste_champs);
mysqli_close($id_connexion);?>
```

## Les bases de données avec MySQL(1)

### ■ Principe des bases de données MySQL

- MySQL s'appuie sur le standard ANSI SQL92
- Par rapport au standard ANSI SQL92, il manque les instructions SELECT imbriquées
- Par rapport au standard ANSI SQL92, il dispose d'un certain nombre de fonctions additionnelles (comparaison de chaînes, expressions régulières, arithmétique sur les dates)
- Le cœur du système MySQL affiche une performance remarquable, tout particulièrement dans les accès indexés
- MySQL utilise une architecture multiutilisateur, multitraitement
  - Permet d'établir des connexions rapides et d'utiliser la même mémoire cache pour plusieurs requêtes



## Les bases de données avec MySQL(2)

### ■ Syntaxe et conventions

- Les chaînes de caractères
  - Notées entre des guillemets simples
  - Notées entre des guillemets doubles dans le cas où la base utilise le mode ANSI
  - Les séquences d'échappement
    - \0 ASCII 0 (NUL)
    - \n Saut de ligne. Correspond au caractère ASCII CHR(13)
    - \t Tabulation. Correspond au caractère ASCII CHR(9)
    - \r Retour chariot. Correspond au caractère ASCII CHR(10)
    - \b Retour caractère
    - \' Guillemet simple
    - \" Guillemet double
    - \\ Barre oblique inverse
    - % Pourcentage
    - \\_ caractère souligné

## Les bases de données avec MySQL(3)

### ■ Syntaxe et conventions

- Les nombres
  - Les nombres à virgule flottante utilisent le point(.) comme séparateur décimal
  - Les valeurs négatives sont précédées du signe moins
  - Les nombres entiers utilisés en relation avec les nombres à virgule flottante sont interprétés comme des nombres à virgule flottante
  - Mysql prend en charge les nombres hexadécimaux
  - NULL signifie « aucune donnée » et ne doit pas être confondu avec le nombre 0 ou la chaîne de caractère vide "" .

## Les bases de données avec MySQL(4)

### ■ Syntaxe et conventions

- Les variables
  - Les variables peuvent contenir des nombres entiers, des nombres réels ou des chaînes de caractères
  - Les noms des variables peuvent comporter des caractères alphanumériques ainsi que les caractères spéciaux . et \_.
  - Il n'est pas nécessaire d'initialiser les variables, qui possèdent par défaut la valeur NULL
  - Syntaxe de Définition de variables
    - SET @variable=(integer | real | string) [,@variable=...]
    - ou
    - @variable:=expression
- Les types de données
  - Mysql connaît les types de données numériques, de date et d'heure ainsi que chaînes

## Les bases de données avec MySQL(5)

### ■ Les types de données

- Les types de données numériques
    - M : nombre maximal de chiffres affichés
    - U (Unsigned) : le caractère de signe est omis
    - Z(Zerofill) : les valeurs manquantes sont remplies par NULL
    - D : le nombre de décimales affichées pour les virgules flottantes
- |                                      |                                                                      |
|--------------------------------------|----------------------------------------------------------------------|
| – TINYINT(M) [UNSIGNED] [ZEROFILL]   | : très petit nombre entier                                           |
| – SMALLINT(M) [UNSIGNED] [ZEROFILL]  | : petit nombre entier                                                |
| – MEDIUMINT(M) [UNSIGNED] [ZEROFILL] | : nombre entier moyen                                                |
| – INT(M) [UNSIGNED] [ZEROFILL]       | : nombre entier normal                                               |
| – INTEGER(M) [UNSIGNED] [ZEROFILL]   | : synonyme de INT                                                    |
| – BIGINT(M) [UNSIGNED] [ZEROFILL]    | : grand nombre entier                                                |
| – FLOAT(precision) [ZEROFILL]        | : nombre à virgule flottante signé                                   |
| – FLOAT(M,D) [ZEROFILL]              | : petit nombre à virgule flottante signé                             |
| – DOUBLE(M,D) [ZEROFILL]             | : petit nombre à virgule flottante normal, à double précision, signé |
| – DOUBLE precision(M,D) [ZEROFILL]   | : synonyme de double                                                 |
| – REAL(M,D) [ZEROFILL]               | : synonyme de double                                                 |
| – DECIMAL(M,D) [ZEROFILL]            | : nombre à virgule flottante signé                                   |
| – NUMERIC(M,D) [ZEROFILL]            | : synonyme de DECIMAL                                                |

## Les bases de données avec MySQL(6)

### ■ Les types de données

#### • Les types de données de date et d'heure

- DATE : Date
- DATETIME : Date et heure
- TIMESTAMP[(M)] : Tampon horaire UNIX
- TIME : Heure
- YEAR[(2|4)] : Année

## Les bases de données avec MySQL(7)

### ■ Les types de données

#### • Les types de données chaînes

- CHAR(M) : chaîne de caractère de longueur fixe M
  - VARCHAR(M) : chaîne de caractère de longueur variable
  - TINYBLOB, TINYTEXT : taille maximale 255 caractères
  - BLOB, TEXT : taille maximale 65535 caractères
  - MEDIUMBLOB, MEDIUMTEXT : taille maximale 16777215 caractères
  - LONGBLOB, LONGTEXT : taille maximale 4294967295 caractères
  - ENUM('value1','value2',...) : Une chaîne de caractères qui n'a qu'une seule valeur, issue d'une liste : ou valeur NULL ou la valeur d'erreur
- encore la spéciale \*\*.
- SET('value1','value2',...) : Une chaîne de caractères qui a zéro, un ou plusieurs valeurs issues d'une liste

## Les bases de données avec MySQL(8)

### ■ Les Opérateurs de MySQL

#### • Les opérateurs arithmétiques

➢ +, -, \*, /

#### • Les opérateurs logiques

➢ NOT (!), OR(||), AND(&&)

#### • Les opérateurs de comparaison

➢ =, <> (!=), <=, <, >=, >

```
Select  NomArt,
        PrixArt AS Prix,
        PrixArt * 0.196 AS 'TVA'
        PrixArt / 10 as 'Marge'
From    article as A,
        groupeArticle as G
Where A.NumGrArt=G.NumGrArt
And    ( G.NumGrArt<>1 OR Not (A.type = 2))
```

## Les bases de données avec MySQL(9)

### ■ Définition et manipulation des données

#### • DDL : langage de définition de données

➢ Définition de tables et de domaine de valeurs

#### • DML : langage de manipulation de données

➢ Modification et sélection d'enregistrement dans les tables

#### • DCL : langage de contrôle de données

➢ Contrôle des accès aux objets et aux opérations, contrôle des transactions

➢ N'est pas pris en charge dans MySQL (version 3.22.32)

## Les bases de données avec MySQL(10)

- Création, suppression et utilisation d'une base de données
  - Création de bases de données
    - CREATE DATABASE nom\_bd
  - Suppression de bases de données
    - DROP DATABASE [IF EXISTS] nom\_bd
      - Le mot-clé IF EXISTS est utilisé pour éviter que la tentative de supprimer une base de données inexistante ne donne lieu à une erreur
  - Utilisation d'une base de données
    - USE nom\_bd
      - Permet d'établir que la base de données spécifiées est la base de données par défaut. Toutes les requêtes suivantes se rapportent donc à cette base de données.

## Les bases de données avec MySQL(11)

- Définition et modification de la structure d'une table
  - Création d'une Table
    - CREATE TABLE : permet de créer une nouvelle table dans la base de données courante
    - Syntaxe :
 

```
--CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name [(create_definition, ...)]
[table_options] [select_statement]
--Create_definition :
*Col_name type [NOT NULL|NULL] [DEFAULT default_value] [AUTO_INCREMENT]
[PRIMARY KEY] [reference_definition] OU
*PRIMARY KEY (index_col_name, ...) OU
*KEY [index_name](index_col_name, ...) OU
*INDEX [index_name](index_col_name, ...) OU
*UNIQUE [INDEX] [index_name] (index_col_name, ...) OU
*[CONSTRAINT symbol] FOREIGN KEY index_name (index_col_name) OU
*[reference_definition]
*CHECK (expr)
```

## Les bases de données avec MySQL(12)

- Définition et modification de la structure d'une table
  - Création d'une Table
    - CREATE TABLE : permet de créer une nouvelle table dans la base de données courante
    - Syntaxe :
      - L'attribut **AUTO\_INCREMENT** : signifie que le contenu d'un champ de type INTEGER est incrémenté automatiquement d'une unité après l'insertion d'un nouvel enregistrement.
        - Ne peut être affecté qu'une seule fois à une table donnée
      - L'attribut **PRIMARY\_KEY** : définit une clé d'index primaire unique
        - Chaque table peut comporter au maximum une clé d'index primaire.
        - Une clé d'index primaire peut être formée à partir d'une combinaison d'un maximum de 32 colonnes
        - Les colonnes d'une clé d'index primaire doivent être définies avec le paramètre NOT NULL

## Les bases de données avec MySQL(13)

- Définition et modification de la structure d'une table
  - Création d'une Table
    - Syntaxe :
      - L'attribut **UNIQUE\_KEY** : définit une clé d'index. Elle ne peut comporter que des valeurs uniques.
      - L'attribut **INDEX** : définit un index.
        - Les champs TEXT et BLOB ne peuvent pas être indexés
      - Les **clés étrangères** : ne sont pas encore prises en charge par MySQL

## Les bases de données avec MySQL(14)

### ■ Définition et modification de la structure d'une table

#### • Création d'une Table

➤ Exemple:

```
CREATE TABLE article (
  NumArt BIGINT NOT NULL AUTO_INCREMENT PRIMARY
  KEY,
  numCde VARCHAR(25) NOT NULL,
  NomART VARCHAR(100) NOT NULL,
  TexteArt MEDIUMTEXT NOT NULL,
  PrixArt DECIMAL(8,2) NOT NULL
  NumGrArt BIGINT NOT NULL,
  NumSGrArt BIGINT NOT NULL,
);
```

## Les bases de données avec MySQL(15)

### ■ Définition et modification de la structure d'une table

#### • Suppression d'une Table

➤ DROP TABLE [IF EXISTS] tbl\_name [, tbl\_name, ...]

#### • Modifier la structure d'une table existante

➤ ALTER [IGNORE] TABLE tbl\_name alter\_spec [, alter\_spec...]

➤ Alter\_specification

```
-ADD [COLUMN] create_definition [FIRST | AFTER column_name]  OU
-ADD PRIMARY KEY (index_col_name, ...)                      OU
-ADD UNIQUE [index_name] (index_col_name, ...)              OU
-CHANGE [COLUMN] old_col_name create_definition            OU
-MODIFY [COLUMN] create_definition                          OU
-DROP [COLUMN] col_name                                     OU
-DROP PRIMARY KEY   OU
-DROP INDEX index_name                                     OU
-RENAME [AS] new_tbl_name                                  OU
-Table_options
---
```

## Les bases de données avec MySQL(16)

### ■ Saisie, modification et suppression d'enregistrement

#### • Insérer de nouveaux enregistrement

```
➤ INSERT INTO article (NumCde, NomArt, TexteArt, PrixArt, NumGrArt,
  NumSGrArt)
  VALUES ('1-001-001-001', 'articl1', 'c'est un article', 7.95, 1, 1);
```

#### • Remplacer un enregistrement

➤ Un ancien enregistrement dans la table, qui a la même valeur qu'un nouvel enregistrement pour une clé, est supprimé de la table avant d'insérer un nouvel enregistrement

```
➤ REPLACE INTO article (NumCde, NomArt, TexteArt, PrixArt, NumGrArt,
  NumSGrArt)
  VALUES ('1-999-999-999', 'articl1', 'c'est un autre article', 12, 1, 1)
```

## Les bases de données avec MySQL(17)

### ■ Saisie, modification et suppression d'enregistrement

#### • Lecture d'enregistrements à partir d'un fichier

```
➤ LOAD DATA LOCAL INFILE 'c:/apache/www3_docs/donnees.txt'
  INTO TABLE article FIELDS TERMINATED BY ',' (NUMCde, NomArt,
  TexteArt, PrixArt, NumGrArt, NumSGrArt);
```

-Les données figurent dans le fichier donnees.txt

-Les données sont séparées par des virgules

-Les données ont \n comme caractère de fin de ligne

-Le mot clé LOCAL indique que le fichier de données se trouve sur l'hôte client dans le répertoire spécifié.

#### • Modification des valeurs dans des champs d'une table

```
➤ UPDATE tbl_name SET col_name=expr1, col_name2=expr2,
  [WHERE section_condition_where]
```

#### • Suppression d'enregistrements

```
➤ DELETE FROM tbl_name [WHERE_definition]
```

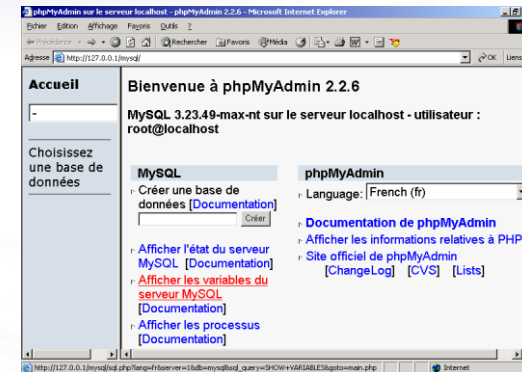
## Les bases de données avec MySQL(18)

### ■ Sélection d'enregistrement

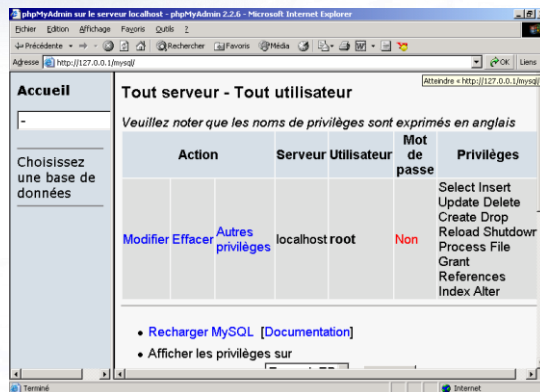
- Syntaxe
 

```
SELECT [DISTINCT|ALL] expression_de_selection
FROM tables
WHERE expression_where
GROUP BY col_name, ...
HAVING where_definition
[ORDER BY [ASC|DESC]]
```
- Exemples
  - `SELECT * FROM article WHERE PrixArt > 50`
  - `SELECT NumGrArt, AVG(PrixArt) FROM article GROUP BY NumArt`

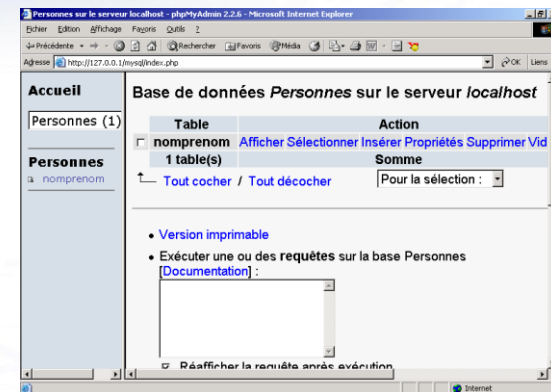
## Les bases de données avec MySQL(19)



## Les bases de données avec MySQL(20)



## Les bases de données avec MySQL(21)



## Les bases de données avec MySQL(22)

Personnes.nomprenom sur le serveur localhost - phpMyAdmin 2.2.6 - Microsoft Internet Explorer

Accueil

Personnes (1)

Personnes

la nomprenom

Base de données *Personnes* - table *nomprenom* sur le serveur *localhost*

[Afficher] [Sélectionner] [Insérer] [Vider] [Supprimer]

Champ	Type	Attributs	Null	Défaut	Extra
<input type="checkbox"/> nom	varchar(20)		Oui	NULL	Modifier Supprimer f
<input type="checkbox"/> prenom	varchar(20)		Oui	NULL	Modifier Supprimer f

↑ Pour la sélection : [Modifier] Ou [Supprimer]

Index : [Documentation]

Type	Espace	Information	Valeur
Données	136 Octets	Format	dynamique
Index	1 024 Octets	Enregistrements	6
Total	1 160 Octets	Longueur enr. ø	22

Aucune clef n'est définie !

Créer une clef

Statistiques :

Type	Espace	Information	Valeur
Données	136 Octets	Format	dynamique
Index	1 024 Octets	Enregistrements	6
Total	1 160 Octets	Longueur enr. ø	22

Taille enr. ø 193 Octets

Internet

http://127.0.0.1/mysql/select.php?lang=fr&server=1&db=Personnes&table=nomprenom&goto=tbl\_properties