PROGRAMMATION Web Dynamique Côté serveur (en PHP)

Zarrouk ELYES

zarrouk.elyes@gmail.com

Plan

- Introduction
- PHP et HTML
- Syntaxe de base de PHP
- L'interactivité en PHP
- Les cookies
- Interfaçage avec les bases de données
- PHP et MySQL
- Les sessions
- Les Mails

Introduction(1)

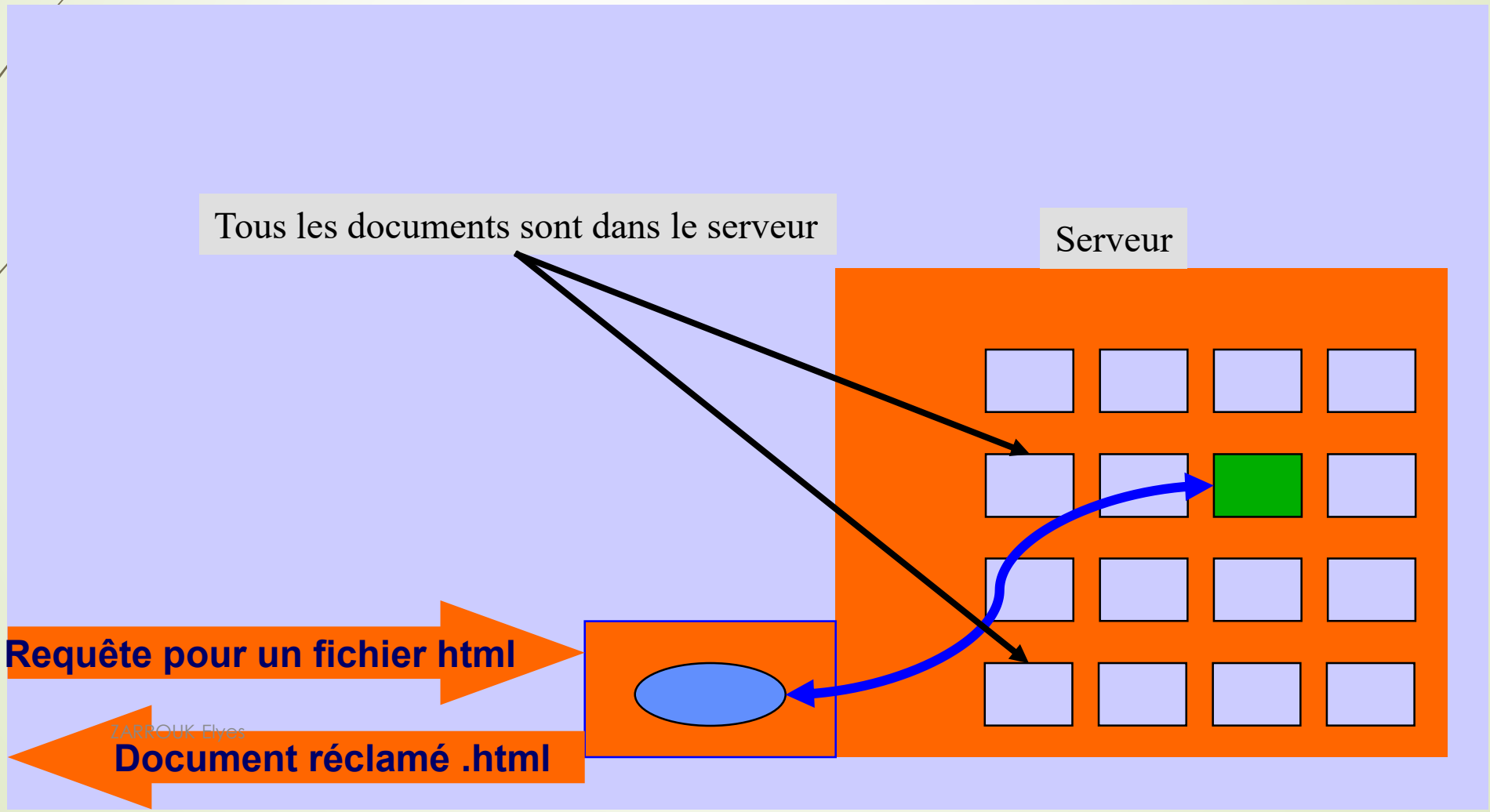
- Internet et les pages web
  - HTML : conception de pages destinées à être publiées sur Internet
  - Page html : contient le texte à afficher et des instructions de mise en page
  - HTML est un langage de description de page et non pas un langage de programmation
    - pas d'instructions de calcul ou 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

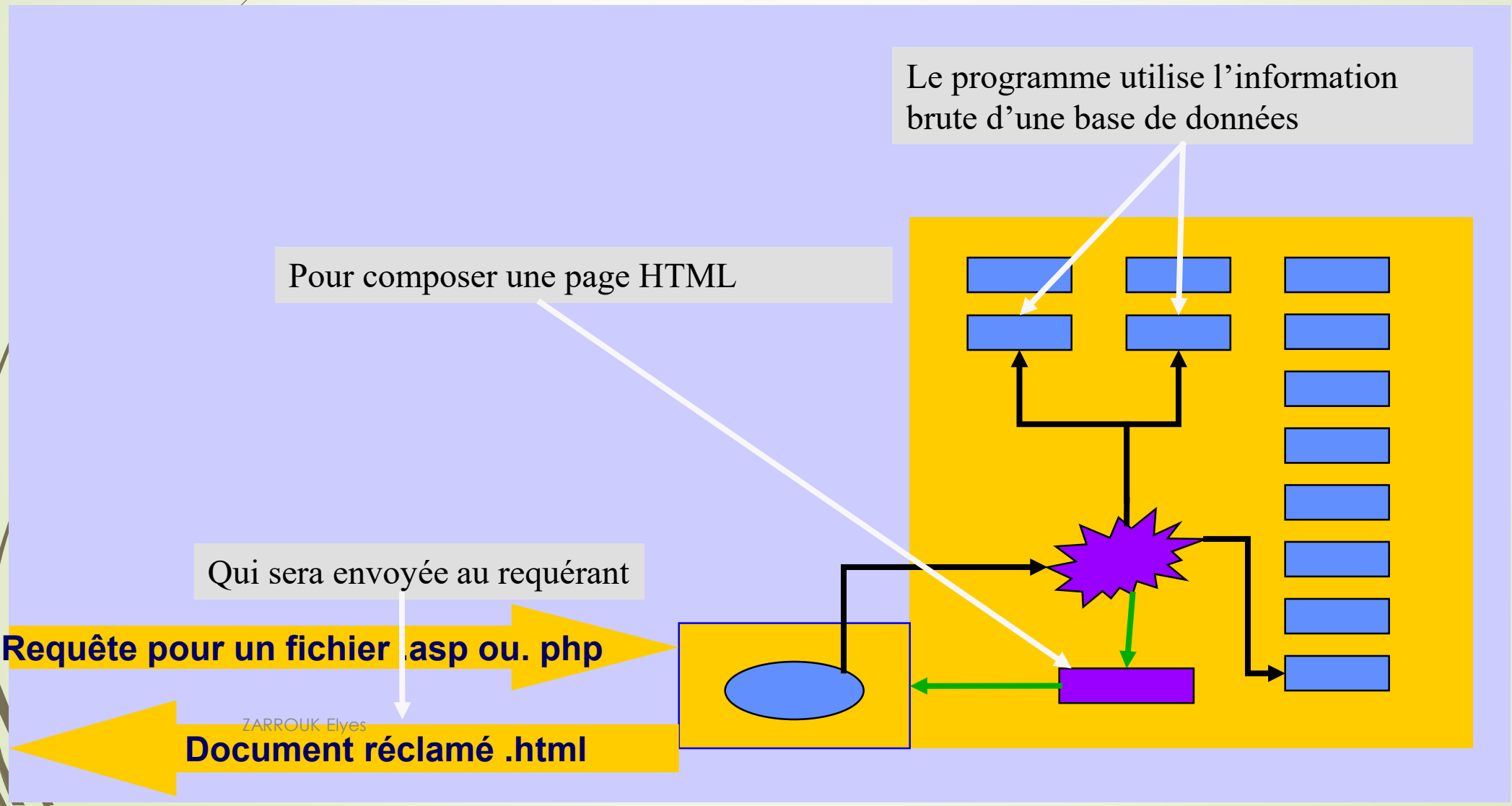
Introduction(3)



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 Netscape 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 traitement 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 (** Computer-generated imagery)
    - 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 (Active Server Pages)**
    - 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 (**Java Server Pages**)**
    - 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 (**Hypertext Preprocessor**)**
    - 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 :
  - 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

Un peu d'histoire

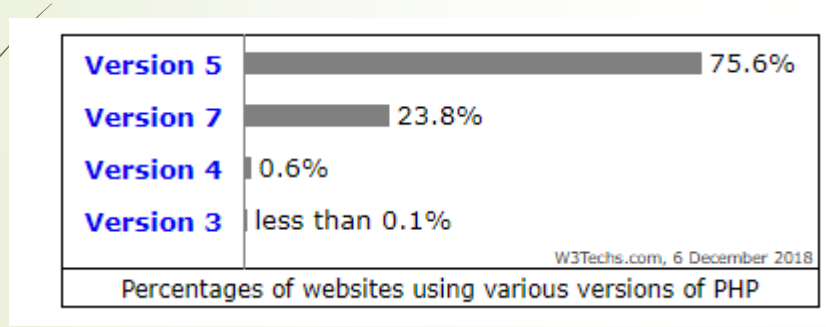
- PHP 5 est sorti en Juillet 2004, après un long développement et plusieurs préversions.
  - Régi par son moteur, le Zend Engine 2.0 avec un nouveau modèle objet
  - 5.4 en 2012, 5.5 en 2013, 5.6 prévue pour l'été 2014
- La version 7.0 de PHP a été annoncée ce jeudi 3 décembre 2015
- PHP 7.0 entre en scène avec un nouveau moteur Zend Engine et des nouvelles fonctionnalités telles que :
  - Prise en charge cohérente du 64 bit
  - Amélioration de la hiérarchie des exceptions ;
  - De nombreuses erreurs fatales converties en exceptions ;
  - Un générateur de nombres aléatoires sécurisé ;
  - Classes anonymes ;

- PHP 7.3 est disponible en version stable

Quelques Chiffres

Année	Part du Marché
2010	75 %
2013	75 %
2016	82 %

En 2010, PHP est le langage dont les logiciels open source sont les plus utilisés dans les entreprises, avec 57 % de taux d'intégration



En 2018, près de 80 % des sites web utilisent le langage PHP sous ses différentes versions

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 xml : <? ligne de code PHP ?>
    - Avec le style php: <?php ligne de code PHP ?>
    - avec le style JavaScript :

<script language=«php»> ligne de code PHP </script>

    - avec le style des ASP : <% ligne de code ASP %>

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 Prinipal

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

Fichier à inclure : information.inc

```
<?php
$a="hello";
$chaine=$salut." C'est PHP ";
echo " <table border= 3
 <tr> <td>
  <h2> $chaine</h2>
  </td> </tr></table> ";
?>
```

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);

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 PHP7”) ;
    - define (“an”, 2019) ;
  - 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(), empty(), gettype(), intval(),
  - is_array(), is_bool(), is_double(), is_float(), is_int(), is_integer, is_long(), is_object(), is_real(), is_numeric(), is_string()
  - Isset(), settype(), strval(), unset()
- 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

$var = 10;//$var=0;
function test()
{
   global $var;
   return $var;
}
$resultat = test();
if ($resultat) {print $resultat;}
else echo " erreur ";

?>
```

➔ 10 //erreur

Syntaxe de base : Les variables(3)

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

```
$nom_variable = 'nom_var';
```

  - Les variables tableaux sont également capables de supporter les noms dynamiques

```
$nom_variable = array("val0", "val1", ...,"valN");
${$nom_variable[0]} = valeur; $val0 = valeur;
$nom_variable = "nom_var";
${$nom_variable}[0] = valeur;
$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)

- Exemple1:

```
<?php
$var = 'hello';
$hello = 'Coucou';
echo $var;
echo ${$var}; ?>
```

- Exemple2:

```
<?php
$tableau1 = array ('test', 'toto', 'titi');
$tableau2 = array ('humpf', 'grmbl');

$var = 'tableau1';
$nb_elements = count (${$var});
for ($i=0; $i<$nb_elements; $i++) {
   // on accede aux éléments du tableau $tableau1
   echo ${$var}[$i].'<br />';
}
?>
```

Syntaxe de base : Les variables (5)

- Variables prédéfinies
  - Les variables d'environnement dépendant du client

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

Syntaxe de base : Les variables (6)

- Variables prédéfinies
  - **Les variables d'environnement dépendant du serveur**

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

Syntaxe de base : Les variables (7)

- Variables prédéfinies
  - Affichage des variables d'environnement
    - la fonction phpinfo()
      - <? phpinfo(); ?>
      - **echo phpinfo(constante);**

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

    - la fonction getenv(constante)
      - <? 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

```
<?      $nbre=10;
if (Settype($nbre, "double")){
echo " la variable $nbre est de type ". gettype($nbre); }?>
```

  - Transtypage explicite : le cast
    - (int), (integer) ; (real), (double), (float); (string); (array); (object)

```
<?php
$v="1025frf";
echo ("pour commencer le type de la variable $v est ".gettype($v));
$v=(double)$v;

echo ("<br>Après le cast, le type de la variable $v est: ".gettype($v));
echo "<br>la valeur est $v";
?>
```

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

```
\tLa nouvelle monnaie unique, l' €uro, 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 = addCSlashes(chaîne, liste_caractères);
ajoute des slashs dans une chaîne

chaîne_result = addSlashes(chaîne);
ajoute un slash devant tous les caractères spéciaux.

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.

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'incrémentation
  - les opérateurs de comparaison
  - les opérateurs logiques
  - les opérateurs bit-à-dit
  - les opérateurs de rotation de bit

Syntaxe de base : Les opérateurs(2)

- Les opérateurs de calcul

  Soit $x=7;

Opérateur	Dénomination	Effet	Exemple	Résultat
+	opérateur d'addition	Ajoute deux valeurs	$x+3	10
-	opérateur de soustraction	Soustrait deux valeurs	$x-3	4
*	opérateur de multiplication	Multiplie deux valeurs	$x*3	21
/	plus: opérateur de 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
%	Modulos	Retourne le reste d'une division	$x % 3	1
**	exponentielle	Valeur1 exposant valeur 2	$x **2	49

Syntaxe de base : Les opérateurs(3)

- Les opérateurs d'assignation

Opérateur	Effet
+=	addition 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 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 OU exclusif 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'incrémentation

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
===	identique	Compare deux valeurs aisni leurs types	$x===$y	Retourne Vrai si $x est égal à $y et les deux sont de même type
<	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

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))
XOR	OU exclusif	Opposé du OU logique	((condition1) XOR (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 (1001 & 1100)	8 (1000)
|	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 (1001 | 1100)	13 (1101)
^	OU bit-à-bit	Retourne 1 si l'un des deux bits de même poids est à 1 (mais pas les deux)	9 ^ 12 (1001 ^ 1100)	5 (0101)
~	Complément (NON)	Retourne 1 si le bit est à 0 (et inversement)	~9 (~1001)	6 (0110)

Syntaxe de base : Les opérateurs(6)

- Les opérateurs de rotation de bit

Opérateur	Dénomination	Effet	Syntaxe	Résultat
<<	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 << 1 (110 <<1)	12 (1100)
>>	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 plus fort est recopié à gauche	6 >> 1 (0110 >>1)	3 (0011)

- Autres opérateurs

Opérateur	Dénomination	Effet	Syntaxe	Résultat
.	Concaténation	Joint deux chaînes bout à bout	"Bonjour"."Au revoir"	"BonjourAu revoir"
$	Référencement de variable	Permet de définir une variable	$MaVariable=2;	
->	Propriété d'un objet	Permet d'accéder aux données membres d'une classe	$MonObjet->Propriete	

Syntaxe de base : Les opérateurs(7)

- Les priorités

Priorité des opérateurs												
	()	[]										
	–	++	!	~	-							
	*	/	%									
	+	-										
	<	<=	>=	>								
	==	!=										
	&											
	^											
	|											
	&&											
	||											
	?	:										
	=	+=	-=	*=	/=	%=	<<=	>>=	>>>=	&=	^=	|=
	AND											
	XOR											

Syntaxe de base : Les instructions conditionnelles(3)

- La boucle for
  - for ($i=1; $i<6; $i++) { echo "$i
"; }
- La boucle while
  - While(condition) {bloc d'instructions ;}
- La boucle do…while
  - Do {bloc d'instructions ;}while(condition) ;
- La boucle foreach
  - Foreach ($tableau as $valeur) {insts utilisant $valeur ;}

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 Valeur1: Liste d'instructions break;
case Valeurs...: Liste d'instructions break;
default: Liste d'instructions break;
}
```

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();
$prod=1;
// la fonction produit a ici $nbarg arguments
for ($i=0; $i <$nbarg; $i++)
{
$prod *= func_get_arg($i);
}
return $prod;
}
echo "le  produit est : ".produit (3, 77, 10, 5, 81, 9)."<br
  />" ;
?>
```

Syntaxe de base : Les fonctions(3)

Les références

- La définition de références permet d'accéder à une zone de mémoire par plusieurs variables
- Une référence se définit comme suit, avec &: $aff_bis = **&**$aff;
- Exemples de manipulation et fonctionnement:
- *Exemple 1 :*

```
$toto = 100;        // la variable $toto est initialisée à la valeur 100
$foobar = &$toto; // la variable $foobar fait référence à $toto
$toto++;            // on change la valeur de $toto
echo $foobar;       // qui est répercutée sur $foobar qui vaut alors 101
```

Syntaxe de base : Les fonctions(4)

- *Exemple 2 :*

```
function change($var) {
   $var++;     // la fonction incrémente en local l'argument
}
$nbr = 1;      // la variable $nbr est initialisée à 1
change(&$nbr);       // passage de la variable par
   référence
echo $nbr;    // sa valeur a donc été modifiée
```

Syntaxe de base : Les fonctions(5)

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

```
<?
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(6)

- 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 actuelle
//récupération des heures et minutes actuelles
$heure = $datejour["hours"];
$minute=$datejour["minutes"];
function bonjour(){
    global $heure;
    global $minute;
    echo "<b> BONJOUR A VOUS IL EST :". $heure. " H ". $minute. "</b> <br  />";}
function bonsoir (){
    global $heure;              global $minute;
    echo "<b> BONSOIR A VOUS IL EST :".$heure. " H ". $minute . "</ b> <br  />";}
if ($heure <= 17)  {$salut = "bonjour";}
          else $salut="bonsoir";
//appel dynamique de la fonction
$salut();?>
```

Syntaxe de base : Les fonctions(7)

- 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 />";   ?>
```

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 indexés 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");
$note = array(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 indexés 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");
&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"; }

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

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

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"));
?>
```

Syntaxe de base : Les tableaux(11)

- 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);

enchaîne des tableaux entrés en argument afin d'en retourner un unique.

$tableau = array_merge_recursive($var_1, $var_2, ..., $var_N);

enchaîne 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 = « Ali" ; $client1-> naiss = "1995" ; $client1->ville =
  "Tunis" ;
//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->Fonction_membre(parametre1,parametre2,...);
  - La variable $this
    - $this->age = $Age;

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

- **Constructeur**

Un constructeur est une méthode spéciale qui est appelée lors de l'instanciation de la classe avec *new*. Le constructeur est affublé d'un nom spécial : ***__construct***. Nous allons donc pouvoir écrire cette méthode de façon à ce qu'elle assigne les valeurs passées en paramètres aux variables membres :

function __construct(…..){……}

- **Destructeur**

Lorsque les objets ne sont plus utilisés, ils sont détruits automatiquement par PHP. Une classe peut posséder un destructeur, qui comme le constructeur est une méthode portant un nom spécial : **__destruct**. A la différence du constructeur, le destructeur ne peut pas accepter de paramètres puisqu'il est appelé automatiquement par Php.

function __destruct() {… }

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

- On va pouvoir définir trois niveaux de visibilité ou d'accessibilité différents pour nos propriétés, méthodes et constants grâce aux mots clefs public, private et protected.
  - ✓ Les propriétés, méthodes ou constantes définies avec le mot clef public vont être accessibles partout, c'est-à-dire depuis l'intérieur ou l'extérieur de la classe.
  - ✓ Les propriétés, méthodes ou constantes définies avec le mot clef private ne vont être accessibles que depuis l'intérieur de la classe qui les a définies.
  - ✓ Les propriétés, méthodes ou constantes définies avec le mot clef protected ne vont être accessibles que depuis l'intérieur de la classe qui les a définies ainsi que depuis les classes qui en héritent ou la classe parente.

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

- **L'héritage**

  - L'héritage permet de faire dériver une classe d'une autre afin qu'elle hérite de tous ses champs de type public et protected ainsi que de toutes ses méthodes

  - 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

  - La 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(5)

- Les opérateurs parent et ::

  - 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 référe 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)

- Exemple:

```
<?php
// Page de définition de la classe trigonometrie.inc
class trigonometrie
{
    var $AB;
    var $AC;

    function hypothenuse()
    {
        $resultat = sqrt(pow($this->AB, 2) + pow($this->AC, 2));
        return $resultat; //number_format($resultat, 2, ',', ' ');
    }
}
?>
```

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

Saisie.php:

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

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

- Resultat.php

```
<?php
include("trigonometrie.inc"); // inclusion de la définition de classe

$trigo = new trigonometrie(); // crée une instance de l'objet
$trigo->AB = $_POST["longueur"];
$trigo->AC = $_POST["hauteur"];
$objet_chaine = serialize($trigo); // sérialise l'objet

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

file_put_contents("fic.txt", $objet_chaine); // écrit l'objet linéarisé dans le fichier

fclose($fichier); // ferme le fichier

/* regroupe tous les éléments du tableau  retourné par la fonction file dans une chaîne */
 $objet_chaine = file_get_contents("fic.txt");
 $trigo = unserialize($objet_chaine); // désérialise l'objet

?>
```

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

- Suite Resultat.php

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

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

- Les fonctions **__sleep** et **__wakeup**
  - Les fonctions **__sleep** et **__wakeup** sont appelées respectivement 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**

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

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

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
<?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.

```
<?php
$cookie_name = "Elyes";
$cookie_value = "Helloooo";
setcookie($cookie_name, $cookie_value, time() + (86400 *
   30), "/"); // 86400 = 1 day
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}?>
```

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
  - 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 »

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

echo " <b> voici le contenu de la chaîne cookie : </b><br>";
echo $name."<br> <br>";
```

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()

```
$exp=explode("&", $name);
echo "<b> ces variables ont été établies à partir de la chaîne
   cookie : </b> <br> <br>";
foreach ($_COOKIE 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 PHP3, si plusieurs cookies sont envoyées de suite, les appels seront traités en ordre inverse, alors qu'avec PHP4 il 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.

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)

- 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(3)


```
<?php

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

Interfaçage avec une base de données(4)


```
        case "ODBC" :  {
                    if ($param_sup == "")
          $id_connexion =  odbc_connect($hote, $utilisateur, $mot_passe);
          else $id_connexion = odbc_connect($hote, $utilisateur, $mot_passe,
                    $param_sup); }
case "Oracle" :   { $id_connexion = ocilogon($utilisateur, $mot_passe); }
case "PostgreSQL" :  { $chaine_connexion = "host=" . $hote . " ";
  if ($port != "") $chaine_connexion .= "port=" . $port . " ";
    $chaine_connexion .= "user=" . $utilisateur ." password=" . $mot_passe";
    if ($param_sup != "") $chaine_connexion .= " ".$param_sup;
  $id_connexion = pg_connect($chaine_connexion);       }
case "Sybase" :       {
   if ($param_sup == "")
    $id_connexion = sybase_connect($hote, $utilisateur, $mot_passe);
  else $id_connexion = sybase_connect($hote, $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(5)

- Php propose 3 API pour se connecter à la Base de données MySQL:
  - **Mysql**
  - **Mysqli**
  - **PDO (PHP Data Objects)**
- La connexion à la base de données est réalisée des fonctions de connexion qui 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(6)

- La connexion à un SGBDR
  - La déconnexion des bases de données s'effectue par l'intermédiaire des fonctions de fermeture
    - mysqli_close($id_connexion);
    - bool mysqli::close();
  - Plusieurs fonctions PHP permettent de retourner des informations à propos de la connexion en cours
    - $chaine_numero_version = **mysqli_get_client_info()**;
    - $type_connexion = **mysqli_get_host_info(**$id_connexion**)**;
    - $protocole_connexion = **mysqli_get_proto_info(**$id_connexion**)**;
    - $chaine_version_serveur = **mysqli_get_server_info(**$id_connexion**)**;

Interfaçage avec une base de données(2)

	ext/mysqli	PDO_MySQL	ext/mysql
PHP version introduced	5.0	5.1	2.0
Included with PHP 5.x	Oui	Oui	Oui
Included with PHP 7.x	Oui	Oui	Non
Development status	Active	Active	removed in 7.x
Recommanded for new projects	**Yes**	**Yes**	**No**

Interfaçage avec une base de données(3)

L'utilisation de MySQL avec PHP s'effectue en 5 temps :

(1) Connexion au serveur de données
(2) Sélection de la base de données
(3) Requête
(4) Exploitation des requêtes
(5) Fermeture de la connexion

- Avec le l'extension MySQL

```
<?php
$c = mysql_connect("example.com", "user", "password");
mysql_select_db("database");
$result = mysql_query("SELECT 'Hello, dear MySQL user!' AS _message
    FROM DUAL");
$row = mysql_fetch_assoc($result);
echo htmlentities($row['_message']);
 ?>
```

Interfaçage avec une base de données(4)

- Avec le l'extension PDO

```
<?php
$pdo = new PDO('mysql:host=example.com;dbname=database', 'user',
    'password');
$statement = $pdo->query("SELECT 'Hello, dear MySQL user!' AS _message FROM
    DUAL");
 $row = $statement->fetch(PDO::FETCH_ASSOC);
echo htmlentities($row['_message']);
```

- Avec le l'extension MySQLi

```
<?php

$mysqli = new mysqli("example.com", "user", "password", "database");
$result = $mysqli->query("SELECT 'Hello, dear MySQL user!' AS _message FROM
    DUAL");
$row = $result->fetch_assoc();
echo htmlentities($row['_message']);
?>
```

Interfaçage avec une base de données(5)

- L'interactivité entre PHP et MySQL peut être gérée à travers deux modes procédurale ou orientée-objet

Connexion à la BDD

- Un style procédural

```
$connexion = mysqli_connect($host,$root,$mdp) or die("erreur");
$con=mysqli_select_db($connexion,$db) or die("Base introuvable");
```

- Un style Orientée Objet

```
$mysqli = new mysqli($host,$root, $mdp, $db);
if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
```

Interfaçage avec une base de données(7)

- 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(8)

- 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(9)

Requête d'insertion

- Un style procédural

```
$connexion = mysqli_connect($host,$root,$mdp) or die("erreur");
$con=mysqli_select_db($connexion,$db) or die("Base introuvable");
```

- Un style Orientée Objet

```
$requete="insert into table values('$val','$val2','$val3','$val4')";
        if (!$mysqli->query($requete))
        {echo "Insertion échouée: (" . $mysqli->errno . ")
```

Ou bien

```
$requete="insert into table values('$val','$val2','$val3','$val4')";
if(!$pdo->query($requete))
        {echo "Insertion échouée";}
```

Interfaçage avec une base de données(10)

Requête de sélection

- Un style procédural

```
$requete="select login from table where Champ='$var'";
        $res = mysqli_query($connexion,$requete);
```

- Un style Orientée Objet

```
$requete="select login from table where Champ='$var'";
$res = $mysqli->query($requete);
```

Ou bien

```
$requete="select login from table where Champ='$var'";
$res= $pdo->query($requete);
```

Interfaçage avec une base de données(11)

- 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 fetch(), 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(12)

Traitement des requêtes

- Un style procédural

```
$requete="select login from table where Champ='$var'";
 $res = mysqli_query($connexion,$requete);
while($row=mysqli_fetch_assoc($res))
{echo $row["login"];}
```

- Un style Orientée Objet

```
 $requete="select login from table where Champ='$var'";
$res = $mysqli->query($requete);
while ($row=$res->fetch_assoc()) {echo $row["login"];}
```

Ou bien

```
 $requete="select login from table where Champ='$var'";
$res= $pdo->query($requete);
while ($row=$res->fetch(PDO::FETCH_ASSOC)){echo $row["login"];}
```

Interfaçage avec une base de données(13)

- 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
    - 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.

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

- numéro et message d'erreur due à la dernière action pour MySQL

```
$num_erreur = mysqli_errno([$id_connexion]);
$message = mysqli_error([$id_connexion]);
```

Interfaçage avec une base de données(14)


```
<?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 {
$id_requete = mysql_query("SELECT datte, email, nom " .
                                        "FROM tbl_utilsateur");
   if (!$id_requete) {
  $message = "<h3>Une erreur est survenue :</h3>" . "<b><u>Erreur numéro " .
   mysqli_errno()
          . ":</u> " . mysqli_error() . "</b>";
    echo $message;
   }}?>
```

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 valeur 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 possède 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 sont remplies par NULL
    - D : le nombre de décimales affichées pour les virgule flottante

- TINYINT[(M)] [UNSIGNED] [ZEROFILL] : très petit nombre entier
- SMALLINT[(M)] [UNSIGNED] [ZEROFILL] : petit nombre entier
- MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL] : nombre entier myen
- INT[(M)] [UNSIGNED] [ZEROFILL] : nombre entier normal
- INTEGER[(M)] [UNSIGNED] [ZEROFILL] : synonyme de INT
- BIGINT[(M)] [UNSIGNED] [ZEROFILL] : grand nombre entier
- FLOAT(précision) [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 précision[(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 4294967295caractères
– ENUM('value1','value2',...) encore la spéciale "".	:Une chaîne de caractères qui n'a qu'une seule valeur, issue d'une liste : ou valeur NULL ou la valeur d'erreur
– 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
    - Sybtaxe :
      - 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
        - PRMARY 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] OU
        - 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 artcile', 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 tebles
    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 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

```
<?php
 session_start();
 $_SESSION["nom_variable"]= valeur

?>
```

  - 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)

- 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 a 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 (7)

- 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();
```

La gestion des connexions aux sites web (1)

- Principe
  - Le langage PHP dispose de plusieurs outils permettant de gérer les connexions des utilisateurs sur un site web
    - Il existe trois états possibles en ce qui concerne le statut des connexions.
      1. **NORMAL** : signifie que la connexion est ouverte et le script est en cours d'exécution.
      2. **ABORTED** : signifie que le client a annulé la connexion et le script est arrêté par défaut.
      3. **TIMEOUT** : signifie que le script a provoqué une déconnexion due à la fin de la durée d'exécution convenue.
    - **Les deux états ABORTED et TIMEOUT peuvent survenir en même temps** dans le cas ou d'une part si PHP est configuré de telle sorte à ignorer les déconnexions et d'autre part lorsque le script arrive à expiration

La gestion des connexions aux sites web(2)

- Principe
  - Le langage PHP dispose de plusieurs outils permettant de gérer les connexions des utilisateurs sur un site web
    - L'état ABORTED en principe interrompt logiquement le script dès que l'utilisateur se déconnecte du site. Toutefois, il est possible de modifier ce comportement en activant l'option de configuration ignore_user_abort dans le fichier php.ini.
    - Si une fonction de fermeture a été enregistrée avec l'instruction register_shutdown_function, le moteur de script se rendra compte après que le client se soit déconnecté puis lors de la prochaine exécution du script que celui ci n'est pas terminé, ce qui provoquera l'appel de la fonction de fermeture mettant fin au script.
    - Lorsque le script se termine normalement, cette fonction sera également appelée.

La gestion des connexions aux sites web (3)

- Principe
  - L'instruction connection_aborted permet d'exécuter une fonction spécifique à la déconnexion d'un client. Si la déconnexion est effective, la fonction connection_aborted retourne true.
  - Un script expire par défaut après une durée de 30 secondes. Néanmoins, ce temps peut être modifié par l'intermédiaire de l'option de configuration max_execution_time dans le fichier php.ini.
  - Une fonction se chargeant de terminer le script sera appelée si le délai arrive à expiration ou si le client se déconnecte.
  - Les fonctions connection_timeout, connection_aborted et connection_status permettent respectivement de vérifier si l'état de la connexion est ABORTED, TIMEOUT et les trois états possibles.

La gestion des connexions aux sites web (4)

- Les fonctions de connexions

connection_aborted();
vérifie si le client a abandonné la connexion.

connection_status();
retourne le statut de la connexion.

connection_timeout();
vérifie l'expiration du script en cours.

ignore_user_abort(paramètre);
détermine si lors de la déconnexion d'un client, le script doit continuer ou arrêter son exécution.

Le paramètre peut valoir soit '0' pour un comportement normal, '1' pour un abandon et '2' pour une expiration.

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(chaîne 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 créé 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 />\n";
}
?>
```

Les en-têtes HTTP (4)

- Les fonctions HTTP

header(chaîne);
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.

Mail (1)

- La fonction mail envoie un message électronique.

Syntaxe :

mail($recipient, $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 en utilisant PHPMailer

```
<?php
require'votre/dossier/PHPMailerAutoload.php';
$mail = new PHPmailer();
$mail->isSMTP(); // Paramétrer le Mailer pour utiliser SMTP
$mail->Host = 'mail.votredomaine.com'; // Spécifier le serveur SMTP
$mail->SMTPAuth = true; // Activer authentication SMTP
$mail->Username = 'user@votredomaine.com'; // Votre adresse email
    d'envoi
$mail->Password = 'secret'; // Le mot de passe de cette adresse email
$mail->SMTPSecure = 'ssl'; // Accepter SSL
$mail->Port = 465;
```

Mail (3)


```
$mail->setFrom('from@example.com', 'Mailer'); // Personnaliser l'envoyeur
$mail->addAddress('To1@example.net', 'Karim User'); // Ajouter le
    destinataire
$mail->addAddress('To2@example.com');
$mail->addReplyTo('info@example.com', 'Information'); // L'adresse de
    réponse
$mail->addCC('cc@example.com');
$mail->addBCC('bcc@example.com');
$mail->addAttachment('/var/tmp/file.tar.gz'); // Ajouter un attachement
$mail->addAttachment('/tmp/image.jpg', 'new.jpg');
$mail->isHTML(true); // Paramétrer le format des emails en HTML ou non

$mail->Subject = 'Here is the subject';
$mail->Body = 'This is the HTML message body';
$mail->AltBody = 'This is the body in plain text for non-HTML mail clients';
```

Mail (4)

Envoyer votre email comme suit :

```
if(!$mail->send()) {
   echo 'Erreur, message non envoyé.';
   echo 'Mailer Error: ' . $mail->ErrorInfo;
} else {
   echo 'Le message a bien été envoyé !';
}
?>
```