

PHP

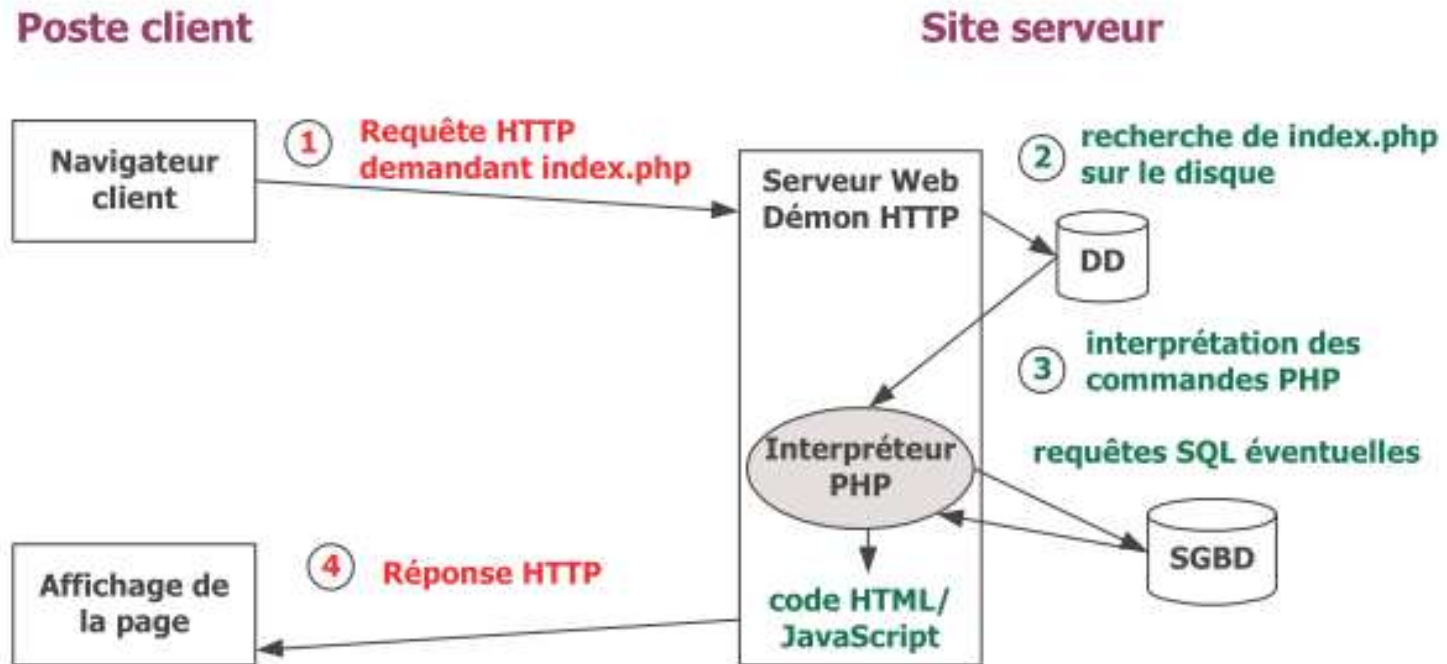




PHP

- PHP (PHP Hypertext Preprocessor) est un langage de script open source côté [serveur](#).
- Langage interprété
- S'exécute coté [serveur](#)
- Le code est intégré au code source de la page HTML
- Permet de rendre les pages HTML Dynamique
- Utilise le protocole [HTTP](#).

PHP





PHP

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 ?>`

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



PHP

On peut intégrer autant de bloc PHP que nous voulons, il suffit d'ouvrir et de fermer le bloc avec `<?php` et `?>`.

```
<?php
    //ligne de code PHP Bloc1
?>
<html>
<head> <title> < ?php
    //ligne de code PHP Bloc 2
?>
    </title> </head>
<body>
    //ligne de code HTML
< ?php
    //ligne de code PHP Bloc 3
?>
    //ligne de code HTML
....
< ?php
    //ligne de code PHP Bloc 4
?>
</body> </html>
```



PHP

Forme d'une page PHP

Intégration « indirecte »

Inclure un fichier PHP dans un fichier HTML : include()

Fichier Principal

```
<html>
<head>
<title> Fichier d'appel </title>
</head>
<body>
<?php
include "information.php" ;
?>
</body>
</html>
```

Fichier à inclure : information.php

```
<?php
echo « Bonjour je suis FLEN» ;
?>
```



Forme d'une page PHP

Intégration « indirecte »

Inclure un fichier PHP dans un fichier HTML : `include()`

Il existe deux fonctions permettant d'inclure un fichier dans un autre :

- `require`
- `include`

Ces deux fonctions jouent le même rôle cependant la différence réside dans leur gestion des erreurs. En effet, si une erreur arrive lors du chargement, `include` génère un avertissement, et le script continue de s'exécuter. Par contre, `require()` génère une erreur fatale et bloque l'exécution du script.

Il existe aussi les variantes `require_once` et `include_once` qui vérifie si le fichier a été déjà ajouté avant si oui il ne l'ajoute plus.



Fonctionnalités de base de PHP

- Commentaires : /* Bloc de commentaires */
// Ligne de commentaire
- Insensible à la casse pour les **fonctions** et **pas pour les noms des variables**
- Toutes les variables ont un nom précédé par **\$**
- Variables non typées à la déclaration (l'affectation détermine le type de la variable)



Fonctionnalités de base de PHP

Lorsqu'on parle de sensibilité à la casse, on parle de faire la différence entre minuscule et majuscule.

PHP est

- Sensible à la casse pour :
 - variables
 - constants
 - clé des tableaux
 - attributs de classes
 - constantes de classes

Non sensible à la casse pour :

- fonctions
- méthodes de classes
- mots clés



Fonctionnalités de base de PHP

- Les variables peuvent être considérés comme des conteneurs de données.
- Les noms de variables commencent avec \$.

Exemple : `$firstname = "aymen";`
`$x=2782;`

- Le nom d'une variable contient uniquement des caractères alphanumérique et le tiret bas (A-z, 0-9, et _).
- Le nom de la variable ne peut commencer que par une lettre ou le tiret bas.
- PHP est un **langage faiblement typé**. Ceci signifie que la variable en PHP épouse le type de son contenu. Si cette variable contient la valeur 5 alors son type est entier. Si on lui affecte ensuite la chaîne "bonjour" elle aura pour type chaîne. L'affectation d'une valeur à une variable détermine son type.

Exemple :

`$x = 2; // $x is integer`

`$x = "hello"; // $x is string`



Fonctionnalités de base de PHP

- Plusieurs fonctions permettent d'afficher une variable en PHP. La plus connue est `echo` qui n'est pas vraiment une fonction et ne se comporte pas comme telle : <http://php.net/manual/fr/function.echo.php>

- Echo permet d'afficher une chaîne de caractères ou tout type convertible en chaîne de caractères.

Petit détour pour les chaînes :

- Lorsque vous utilisez `"`, vous pouvez insérer directement des variables dans votre chaîne de caractères. Si vous avez donc une variable `$x = "aymen"`, alors `echo " Bonjour $x";` sera affiché : Bonjour aymen. En effet, php va chercher dans la chaîne la présence de variable et va les interpréter avant d'afficher la chaîne. On peut aussi écrire `echo " Bonjour {$x}";`
- Les guillemets simples annulent quant à elles le remplacement des variables. `echo ' Bonjour $x';` sera affiché : Bonjour \$x.
- Dans le cas de guillemets simples, nous concaténons avec l'opérateur `.`.
`echo 'Bonjour' . $x; //` Bonjour aymen
- Il existe d'autres façons d'afficher des variables avec `print` ou `printf`. Faites une petite recherche et testez ces différentes façons.



Fonctionnalités de base de PHP

- Une **variable dynamique** est une variable qui permet d'avoir un **nom dynamique d'une variable**. Elle prend la valeur d'une variable et l'utilise comme nom d'une autre variable.
- Pour déclarer une variable dynamique on utilise **\$\$**.
- Exemple :
`$ndv = 'varDyn'`
`$$ndv = 'contenuVarDyn';`

Quels sont les variables présentes dans ce script ?

Ici nous avons deux variables **\$ndv** qui contient la chaîne **varDyn** et **\$varDyn** qui contient la chaîne **contenuVarDyn**



Fonctionnalités de base de PHP

- Les principaux types de PHP (non exhaustifs) sont :
 - Les chaînes de caractères (String en anglais) qui permettent de stocker du texte. Elle sont écrites entre guillemets ou entre apostrophes.
 - Les nombres
 - Les réels
 - Les booléens : qui a pour valeur `true` ou `false` et qui permet d'évaluer une expression



Fonctionnalités de base de PHP

- Le nom de votre variable doit être significatif. Eviter les abréviations comme pers pour personne.
- Une variable peut être une left value ou **lvalue**. Ceci s'applique lorsque cette variable se trouve à gauche d'une affectation. Par exemple dans l'instruction `$x = 2;` la variable `$x` est une **lvalue**, elle joue le rôle d'un récipient dans lequel on va stocker le contenu de la partie droite.
- Lorsque la variable se trouve à droite d'une opération, elle est interprétée et on la remplace par son contenu.

Exemple;

```
$x= 2; //lvalue $x est un récipient  
$z=$x + 3; // Ici $z est une lvalue, par contre $x est remplacé par sa  
           // valeur 2 et on aura finalement dans $z la valeur 5.
```



Fonctionnalités de base de PHP

➤ Variable locale

Visible uniquement à l'intérieur d'un contexte d'utilisation

➤ Variable globale

Visible dans tout le script

Accessible localement avec l'instruction `global` ou avec la variable globale `$GLOBALS`

Exemple :

```
<?php
$globalVar1 = 1;
$globalVar2 = 2;
function somme() {
    global $globalVar1, $globalVar2;
    $globalVar2 = $globalVar1 +
    $globalVar2;
}
somme();
echo $ globalVar2;
```

```
<?php
$globalVar1 = 1;
$globalVar2 = 2;
function somme() {
    $GLOBALS['globalVar2'] =
    $GLOBALS['globalVar1'] +
    $GLOBALS['globalVar2'];
}
somme();
echo $ globalVar2;
?>
```



Fonctionnalités de base de PHP

➤ Fonctions de vérifications de variables

floatval() : Convertit une chaîne en nombre à virgule flottante

empty() : Détermine si une variable est vide

gettype() : Retourne le type de la variable (boolean, integer, double(float), string, array, object, null, unknown type)

intval() : Retourne la valeur numérique entière équivalente d'une variable

is_Type() : () , is_bool(), is_double(), is_float(), is_int(), is_integer, is_long(), is_obj, is_array(), is_real(), is_numeric(), is_string() vérifie si la variable est de type **Type**.

isset() : Détermine si une variable est définie et est différente de NULL

settype(var, newType) : Affecte un type à une variable

strval() : Récupère la valeur d'une variable, au format chaîne

unset() : Détruit une variable

var_dump() : Affiche les informations d'une variable



Fonctionnalités de base de PHP

➤ Variables d'environnement 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
<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
<code>\$_SERVER["PHP_SELF"]</code>	Nom du script PHP



Fonctionnalités de base de PHP

➤ Variables d'environnement 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

`phpinfo(INFO_VARIABLES);` permet d'afficher les variables d'environnement



Fonctionnalités de base de PHP

- Une constante est un identifiant (un nom) qui représente une valeur simple.
- Une constante ne peut jamais être modifiée durant l'exécution du script.
- Par convention, les constantes sont toujours en majuscules.
- Pour définir une constante dans un script et hors d'une classe PHP on utilise la syntaxe suivante :
`define("nomDeLaCostante", "valeurDeLaConstante");`
- Exemple :
`define("jeSuisUneConstante", "bonjour");`



Fonctionnalités de base de PHP

PHP propose aussi un grand nombre de constante prédéfinies. Etant données que le nombre est trs grand, vous pouvez lister ces variable en utilisant la fonction : `get_defined_constants()` qui liste l'ensemble de ces fonctions.

```
<?php
    print_r(get_defined_constants());
?>
```

Nous pouvons voir par exemple la constante `__FILE__` qui informe sur le nom du fichier en cours d'exécution.

`PHP_Version` qui donne la version PHP utilisée ou encore `PHP_OS` qui informe sur l'OS utilisé dans le serveur.



Fonctionnalités de base de PHP

Opérations élémentaires

Exemple	Nom	Résultat
$+$a$	Identité	Conversion de $$a$ vers int ou float , selon le plus approprié.
$-$a$	Négation	Opposé de $$a$.
$$a + b	Addition	Somme de $$a$ et $$b$.
$$a - b	Soustraction	Différence de $$a$ et $$b$.
$$a * b	Multiplication	Produit de $$a$ et $$b$.
$$a / b	Division	Quotient de $$a$ et $$b$.
$$a \% b	Modulus	Reste de $$a$ divisé par $$b$.
$$a ** b	Exponentielle	Résultat de l'élévation de $$a$ à la puissance $$b$. Introduit en PHP 5.6.



Fonctionnalités de base de PHP

- 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



Fonctionnalités de base de PHP

```
if($temperature < 0) {  
    echo 'il fait très froid';  
}elseif ($temperature<10){  
    echo 'la température est gérable';  
}elseif ($temperature<26){  
    echo 'la température est agréable';  
}else{  
    echo 'il fait chaud';  
}
```



Fonctionnalités de base de PHP

- Une nouvelle variante proposée par PHP7 de La condition ternaire `?:` est l'opérateur `??`, il permet généralement d'affecter des **valeur par défaut** une variable si la première affectation ne fonctionne pas.
- Syntaxe : `$var = $newV1 ?? $newV2 ?? ... ?? $newVn`

Prenons cet exemple

- `$maVariable = isset($newVal1)? $newVal1 : $newVal2;`
Ce code va vérifier si `$newVal1` existe et non null, c'est le travail de la fonction `isset`. Si elle existe alors il l'affectera à `$maVariable` sinon il affectera la variable `$newVal2`.

Afin de raccourcir ce traitement qui est assez récurrent on utilise la syntaxe suivante :

`$var = $newVal1 ?? $newVal2;`

On peut enchaîner plusieurs fois et le raisonnement reste le même.



Fonctionnalités de base de PHP

L'instruction **switch** permet de compresser un code avec if et elseif. Ici on teste sur la valeur de Variable et selon (case) la valeur qu'elle prend on exécute un bloc d'instructions.

Switch ne teste que l'égalité.

Le mot clé break permet d'arrêter le traitement une fois les instructions du bloc sélectionné se terminent.

Sans le mot clé break, tous les instructions qui suivent seront exécutées.

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

```
$x=2;  
switch ($x) {  
    case 1 : echo 'Lundi';  
        break;  
    case 3 : echo 'Mercredi';  
        break;  
    default :  
        echo 'Autre jour';  
}
```



Fonctionnalités de base de PHP

➤ Lorsque un traitement se répète plusieurs fois, il est nécessaire d'utiliser les structures itératives.

➤ La boucle for

La syntaxe de la boucle for est la même qu'en C ou en Javascript.

```
for (initialisation; Condition ; incrémentation) {  
    Block d'instructions;  
}
```

Exemple : Affichage des chiffres de 1 à 10, chacun dans une ligne

```
for ($i=1; $i<=10 ; $i++) {  
    echo $i. '<br>';  
}
```

Exercice

- Créer une page PHP qui permet d'afficher la table de multiplication Donnant la figure suivante :

Ps : N'oublier pas de commenter votre code

X	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81



Fonctionnalités de base de PHP

- Les boucles while et do While

Elles permettent de répéter le bloc d'instruction tant que la condition est vrai.

- `While (condition) {`
 bloc d'instructions ;
 }

- `do {`
 bloc d'instructions ;
 } `While (condition);`

- La différence entre ces deux boucles est que la première peut n'exécuter aucune instruction. Pour la seconde au moins une instruction est exécutée.

```
while ($x%2 != 0) {  
    echo $x. ' n'est pas divisible par deux<br>';  
    $x=($x*3) - 1;  
}
```



Exercice

- 1- Regarder dans la documentation comment générer un nombre aléatoire avec PHP.
- 2- Effectuer une suite de tirages de 3 nombres aléatoires jusqu'à obtenir une suite composée d'un nombre pair suivi de deux nombres impairs.



Fonctionnalités de base de PHP

➤ Une chaîne de caractère est une série de caractères.

Quelques fonctions prédéfinies pour la gestion de chaînes de caractères

Pour les détails sur ces fonctions et d'autres fonctions de chaînes consultez le Manuel PHP : <http://php.net/manual/fr/ref.strings.php>

[chr](#) — Retourne un caractère à partir de son code ASCII

[count_chars](#) — Retourne des statistiques sur les caractères utilisés dans une chaîne

[echo](#) — Affiche une chaîne de caractères

[explode](#) — Coupe une chaîne en segments et prends en paramètres le délimiteur et la chaîne.

[fprintf](#) — Écrit une chaîne formatée dans un flux

[htmlentities](#) — Convertit tous les caractères éligibles en entités HTML

[htmlspecialchars_decode](#) — Convertit les entités HTML spéciales en caractères

[htmlspecialchars](#) — Convertit les caractères spéciaux en entités HTML

[implode](#) — Rassemble les éléments d'un tableau en une chaîne

[lcfirst](#) — Met le premier caractère en minuscule

[ltrim](#) — Supprime les espaces (ou d'autres caractères) de début de chaîne

[money_format](#) — Met un nombre au format monétaire

[nl2br](#) — Insère un retour à la ligne HTML à chaque nouvelle ligne

[ord](#) — Retourne le code ASCII d'un caractère

[print](#) — Affiche une chaîne de caractères



Fonctionnalités de base de PHP

- [printf](#) — Affiche une chaîne de caractères formatée
- [rtrim](#) — Supprime les espaces (ou d'autres caractères) de fin de chaîne
- [sprintf](#) — Retourne une chaîne formatée
- [scanf](#) — Analyse une chaîne à l'aide d'un format
- [str_getcsv](#) — Analyse une chaîne de caractères CSV dans un tableau
- [str_repeat](#) — Répète une chaîne
- [str_replace](#) — Remplace toutes les occurrences dans une chaîne
- [str_shuffle](#) — Mélange les caractères d'une chaîne de caractères
- [str_split](#) — Convertit une chaîne de caractères en tableau
- [str_word_count](#) — Compte le nombre de mots utilisés dans une chaîne
- [strcmp](#) — Comparaison binaire de chaînes
- [strip_tags](#) — Supprime les balises HTML et PHP d'une chaîne
- [stripos](#) — Recherche la position de la première occurrence dans une chaîne, sans tenir compte de la casse
- [stripslashes](#) — Supprime les antislashes d'une chaîne
- [strstr](#) — Trouve la première occurrence dans une chaîne
- [stristr](#) — Version insensible à la casse de strstr
- [strlen](#) — Calcule la taille d'une chaîne
- [strncmp](#) — Comparaison binaire des n premiers caractères



Fonctionnalités de base de PHP

[strpos](#) — Cherche la position de la première occurrence dans une chaîne

[strrchr](#) — Trouve la dernière occurrence d'un caractère dans une chaîne

[strrev](#) — Inverse une chaîne

[stripos](#) — Cherche la position de la dernière occurrence d'une chaîne contenue dans une autre, de façon insensible à la casse

[strrpos](#) — Cherche la position de la dernière occurrence d'une sous-chaîne dans une chaîne

[strtok](#) — Coupe une chaîne en segments

[strtolower](#) — Renvoie une chaîne en minuscules

[strtoupper](#) — Renvoie une chaîne en majuscules

[strtr](#) — Remplace des caractères dans une chaîne

[substr_compare](#) — Compare deux chaînes depuis un offset jusqu'à une longueur en caractères

[substr_count](#) — Compte le nombre d'occurrences de segments dans une chaîne

[substr_replace](#) — Remplace un segment dans une chaîne

[substr](#) — Retourne un segment de chaîne

[trim](#) — Supprime les espaces (ou d'autres caractères) en début et fin de chaîne

[ucfirst](#) — Met le premier caractère en majuscule

[ucwords](#) — Met en majuscule la première lettre de tous les mots



Fonctionnalités de base de PHP

Les tableaux

Exercice

Ecrire une fonction qui permet de transformer une chaîne de caractère du format jj/mm/aaaa vers jj-mm-aaaa.



Fonctionnalités de base de PHP

- Un tableau est une succession d'éléments de **différents types**
- Deux méthodes permettent de créer un tableau :
 - En affectant à une variable un tableau en utilisant les **[]**
`$tab = [];` // crée un tableau vide
`$tab = [1,'abc'];` // crée un tableau avec deux éléments
 - En utilisant la fonction **array()**
`$tab1 = array();` // crée un tableau vide
`$tab1 = array(1,'abc');` // crée un tableau avec deux éléments
- Les éléments d'un tableau peuvent **pointer vers d'autres tableaux**
- **L'index** d'un tableau en PHP commence de **0**
- On **ne définit pas la taille** du tableau
- Pour **ajouter/modifier** un élément dans un tableau on utilise la syntaxe suivante : **tab[indice] = valeur.**
- Pour supprimer un élément d'un tableau on utilise la fonction **unset** qui prend en **paramètre l'élément supprimer.**
- La fonction **count()** pour avoir le nombre d'éléments d'un tableau
- Il existe deux types de tableaux
 - Tableau indicés
 - Tableau associatif



Fonctionnalités de base de PHP

- Tableau **indicés** : Les éléments sont accessible par leur index qui commence par 0 et en utilisant les []. Exemple t[5] affiche l'élément d'indice 5.

Exemple

```
$cartes = array(1,2,3,4,5,6,7,8,9,10,11,12,13);
```

Parcours

```
echo "Affichage avec for : <br>";
```

```
for ($i=0;$i<count($cartes);$i++){  
    echo "carte de valeur : ". $cartes[$i]. "<br>";  
}
```

```
echo "Affichage avec foreach : <br>";
```

```
foreach($cartes as $carte){  
    echo "carte de valeur : ". $carte. "<br>";  
}
```



Fonctionnalités de base de PHP

Les tableaux

- Tableau **associatif** : L'indice de chaque élément est une chaîne de caractère.

Exemple

```
$cartes = array(  
    « As »=>1, « Dous »=>2, « Tris »=>3, « Quatro »=>4, « Chinka »=>5,  
    « six »=>6, « Sept »=>7, « huit »=>8, « neuf »=>9, « Dix »=>10, « Valet »=>11,  
    « Dame »=>12, « Roi »=>13,  
);
```

Parcours

```
foreach($cartes as $indCarte => $Carte) {  
    echo "La carte ". $indCarte . " de valeur : " . $Carte . "<br>";  
}
```

<http://php.net/manual/fr/language.types.array.php>

Fonctionnalités de base de PHP

Exercice

Afficher en utilisant la fonction `get_defined_constants()` qui liste l'ensemble des constantes prédéfinies l'ensemble de ces valeurs dans un tableau comme illustré dans cette figure

Nom Constante	Valeur
E_ERROR	1
E_RECOVERABLE_ERROR	4096
E_WARNING	2
E_PARSE	4
E_NOTICE	8
E_STRICT	2048
E_DEPRECATED	8192
E_CORE_ERROR	16
E_CORE_WARNING	32
E_COMPILE_ERROR	64
E_COMPILE_WARNING	128
E_USER_ERROR	256
E_USER_WARNING	512
E_USER_NOTICE	1024
E_USER_DEPRECATED	16384



Fonctionnalités de base de PHP

Les tableaux

Exercice

En utilisant les deux fonctions `count_chars($chaine,1)` qui retourne le nombre d'occurrence de chaque caractère dans la chaîne entrée en paramètre et la fonction `chr` qui retourne le caractère du code ASCII introduit en paramètre, afficher chaque caractère suivi de son nombre d'occurrence.



Fonctionnalités de base de PHP

- L'une des fonctionnalité la plus utile dans un tableau est la recherche. Nous pouvons chercher sur deux éléments :
 - Les **clés** avec la fonction **array_key_exists** qui prend en paramètre la clé et le tableau et retourne un booléen indiquant si le tableau contient cette clé ou pas.
 - Les **valeurs** avec la fonction **in_array** qui prend aussi en paramètre l'élément à chercher et le tableau et retourne un booléen indiquant si le tableau contient cet élément ou pas.
 - Nous pouvons aussi récupérer la clé d'un élément dans un tableau s'il existe avec la fonction **array_search**. Si la valeur n'existe pas dans le tableau la fonction renvoi **false**.



Fonctionnalités de base de PHP

- `$tableau = array_count_values($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_merge($var_1, $var_2, ..., $var_N)` enchaîne des tableaux entrés en argument afin d'en retourner un unique.



Fonctionnalités de base de PHP

- `$tableau = array_merge_recursive($var_1, $var_2, ..., $var_N)` enchaîne des tableaux en conservant l'ordre des éléments dans le tableau résultant.
- `sort($var)` : tri les valeurs du tableau selon le code ASCII
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.
- `rsort ($var)` tri en ordre inverse des codes ASCII.
- `asort ($var)` trie également les valeurs selon le critère des codes ASCII, mais en préservant les clés pour les tableaux associatifs
- `arsort ($var)` la même action mais en ordre inverse des codes ASCII
- `natscasesort ($var)` effectue un tri dans l'ordre alphabétique non ASCII (« a » est avant « z » et « 10 » est après « 9 »)



Fonctionnalités de base de PHP

- `sort($var, $flag)` : tri les valeurs du tableau selon type introduit par `$flag`. Le tableau `initial` est `modifié et non récupérable` dans son ordre original. Pour les tableaux `associatifs` les `clés seront perdues` et remplacées par un indice créé après le tri.

Voici quelques exemples de la valeur que peut avoir `$flag` :

- `SORT_REGULAR` (la valeur par défaut donc si vous ne mettez rien c'est cette valeur qui est prise en considération) : compare les éléments normalement (ne modifie pas les types)
- `SORT_NUMERIC` : compare les éléments numériquement
- `SORT_NATURAL` - compare les éléments comme des chaînes de caractères en utilisant l'ordre naturel comme le fait la fonction

<http://php.net/manual/fr/function.sort.php>



Fonctionnalités de base de PHP

- Tri personnalisé des tableaux associatif
 - Sachant que le tri utilise le code ASCII pour comparé les éléments, on peut personnaliser la fonction de comparaison de la manière suivante.

```
function compare($a, $b) {  
    if ($a == $b) {  
        return 0;  
    }  
    return ($a < $b) ? -1 : 1;  
}
```

```
// Tableau à trier  
$array = array('a' => 4, 'b' => 8, 'c' => -1,  
              'd' => -9, 'e' => 2, 'f' => 5, 'g' => 3, 'h' => -4);  
print_r($array);
```

```
// Trie et affiche le tableau résultant  
uasort($array, 'compare');
```



Fonctionnalités de base de PHP

Date

➤ Plusieurs fonctions de gestion des dates.

`getdate()` : Retourne un tableau associatif contenant les informations de date et d'heure du timestamp lorsqu'il est fourni, sinon, le timestamp de la date/heure courante locale.

<http://php.net/manual/fr/function.getdate.php>

`date()` : Retourne une date sous forme d'une chaîne, au format donné par le paramètre format, fournie par le paramètre timestamp ou la date et l'heure courantes si aucun timestamp n'est fourni. En d'autres termes, le paramètre timestamp est optionnel et vaut par défaut la valeur de la fonction `time()`.

<http://php.net/manual/fr/function.date.php>



Fonctionnalités de base de PHP

➤ Déclaration et appel d'une fonction

```
Function nom_fonction($arg1, $arg2, ...$argn)
{
    déclaration des variables ;
    bloc d'instructions ;
    // en cas de retour de valeur
    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



Fonctionnalités de base de PHP

Les fonctions

- Exemple fonction avec nombre d'arguments inconnu

Ecrire une fonction qui calcule le produit des paramètres qui lui ont passé en argument. Le 0 n'est pas comptabilisé

```
<?php
function produit()
{
    $nbarg = func_num_args() ;
    $prod=1 ;
    for ($i=0 ; $i <$nbarg ; $i++)
    {
        $prod *= func_get_arg($i) ;
    }
    return $prod;
}
echo "le produit est : ", produit (2, 7, 82, 8, 11, 2016), "<br />" ;
?>
```



Fonctionnalités de base de PHP

Avec PHP 7 un nouvel opérateur est apparu comme alternative de la fonction `func_get_args()`. C'est l'opérateur `...`

Reprenons la fonction `somme` et ajoutons cet opérateur :

```
function somme (...$args) { }
```

Nous récupérons maintenant directement dans le tableau que nous avons appelé `$args` l'ensemble des paramètres passé à la fonction sans avoir à utiliser l'ancienne méthode.



Fonctionnalités de base de PHP

➤ Exercice

Ecrire une fonction produit qui effectue le produit sur un nombre inconnue de paramètres. Si l'un des paramètres est 0 on ne le comptabilise pas.



Fonctionnalités de base de PHP

Les fonctions

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

<?

Ecrire une fonction `sommeProduit` qui calcule la somme et le produit de deux entiers passés en paramètre

```
function sommeProduit(&$som, &$prod,$x,$y)
{
    $som=$x+$y;
    $prod=$x*$y;
}
$s=0;
$p=0;
sommeProduit($s,$p,5,2);
```

```
echo " somme = $s et prod = $p";
```

?>

- L'appel récursif
 - PHP admet les appels récursifs de fonctions



Exercice

Ecrire une fonction qui prend en entrée un tableau d'entiers et le décompose en deux tableaux, l'un contenant les éléments pairs et l'autre les éléments impaires. La fonction aura donc comme paramètres les trois tableaux.



Fonctionnalités de base de PHP

Il existe deux types de typages en PHP :

1. Typage Faible

Dans ce type de typage, en cas où le type de la variable ne correspond pas, un transtypage est alors automatiquement exécuté. Par exemple si on reprend l'exemple de la fonction somme

```
function somme(int $x, int $y){return $x + $y;}
```

Supposons que nous avons l'appel suivant : somme(2.5,3) alors la fonction s'exécutera avec les valeurs 2 et 3. 2 étant le transtypage de 2.5.

2. Typage Fort

Appelé aussi typage strict. Ce typage refuse tout paramètre dont le type diffère du type attendu. Appliqué lorsqu'on ajoute en haut du script avec l'appel suivant : declare(strict_types=1).

On peut aussi avoir un typage sur la valeur de retour.

Exemple :

```
int function somme(int $x, int $y){return $x + $y;}
```



Fonctionnalités de base de PHP

Les paramètres par défaut de la fonction

On peut ajouter des paramètres par défaut aux paramètres de la fonction afin de les prendre en considération en cas où l'utilisateur ne passe pas les paramètres attendus.

Exemple

```
int function somme(int $x=0, int $y=0) {  
    return $x + $y;  
}
```

Dans cet exemple, si on exécute cet appel `somme()`;

La fonction n'ayant pas reçu de paramètres va utiliser les paramètres par défaut. `$x` et `$y` prendront les valeurs 0. Donc la fonction retournera 0.



Fonctionnalités de base de PHP

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



Fonctionnalités de base de PHP

Typage des paramètres

Depuis PHP 7, il est possible de définir un typage des paramètres et surtout un typage scalaire.

Les types possible sont :

1. Int
2. float
3. string
4. bool
5. array
6. Nom de classe
7. Nom d'une interface

La fonction somme par exemple devient la suivante :

```
function somme(int $x, int $y) {  
    return $x + $y;  
}
```



Fonctionnalités de base de PHP

Les formulaires sont un maillon essentiel et incontournable permettant l'interaction entre un site ou une application web et ses utilisateurs.

Revenons un peu sur les éléments de bases à maîtriser dans un formulaire pour permettre cet échange.

La balise permettant de créer un formulaire et la balise form :

`<form>` `</form>` nous permet donc de délimiter le formulaire. Les attributs associé à cette balise sont :

action : c'est le fichier dans le serveur qui va traiter les informations saisies.

Remarque : Si vous voulez que le fichier qui contient le formulaire soit celui qui le traite, garder le champ **action vide** ou utiliser la variable super globale `$_SERVER` et accéder à la variable `PHP_SELF` : `$_SERVER["PHP_SELF"]`.



Fonctionnalités de base de PHP

method : qui prend ou la valeur **post** ou la valeur **get** et qui détermine la d'envoi des données vers le serveur. La méthode **get**, qui est la méthode par défaut, présente l'inconvénient d'ajouter les données du formulaire à l'adresse URI du fichier qui les traite, ce qui les rend visibles par le visiteur. Cet inconvénient peut être exploité pour passer des données à un script dès son appel. De plus, il existe une limite à la longueur des URI et donc à la quantité de données à transmettre. Ces problèmes ne se retrouvent pas avec la valeur **post**.

name : nom du formulaire, utile surtout en JS.

enctype : identifie le type d'encodage des données transmises au serveur. Généralement ce champ n'est pas ajouté et il prend donc sa valeur par défaut ("**application/x-www-form-urlencoded**"). En cas d'envoi de fichier du client vers le serveur (upload image par exemple), l'encodage doit être "**multipart/form-data**".



Fonctionnalités de base de PHP

- Afin de récupérer une variable envoyée à travers un formulaire, nous utilisons les variables globales `$_POST` et `$_GET` selon la méthode utilisée lors de l'envoi et le nom de la variable envoyée (le contenu de l'attribut `name` qui est **OBLIGATOIRE**).
- Lorsque vous utilisez un champ file pour envoyer des fichiers vers le serveur, vous devez utiliser la variable `$_FILES` qui utilise les mêmes règles que post et get pour l'accès à une donnée.
- Ces variables sont des tableaux associatifs.
- Exemple :
 - Pour récupérer la variable de nom « nom » et envoyée par post on utilise : `$_POST['nom'];`



Exemple

Soit la page HTML suivante qui permet d'envoyer le nom de l'utilisateur à la page Bonjour.php

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
<form action="bonjour.php" method="post">
  <input type="text" name="nom">
  <input type="submit" value="envoyer">
</form>
</body>
</html>
```



Exemple

La page « bonjour.php » va recevoir les données envoyées par la page bonjour.html dans la variable `$_POST`. Ensuite elle va dire bonjour au nom envoyé par le formulaire

```
<html>
<head>
    <title>Bonjour</title>
</head>

<body>
    Bonjour <?= $_POST[ 'nom' ] ?>
</body>
</html>
```



Exemple

Modifier le code du formulaire en changeant la méthode de post vers get.

Envoyer votre requête et vérifier ce qui se passe au niveau de l'URL.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<form action="bonjour.php" method="get">
    <input type="text" name="nom">
    <input type="submit" value="envoyer">
</form>
</body>
</html>
```

<http://localhost/gmc/bonjour.php?nom=aymen>



Exemple

Si vous avez seulement modifié la méthode post vous allez avoir cet affichage



Bonjour

Notice: Undefined index: nom in C:\xampp\htdocs\gmc\bonjour.php on line 18

La première constatation est que l'URL contient le paramètre envoyé. C'est la spécificité du `get` et ça ne marche pas qu'avec les formulaires en effet vous pouvez créer votre URL et y injecter des données en utilisant cette syntaxe.

La seconde constatation est l'erreur qui apparaît et c'est normal vu que vous n'avez pas modifié la variable super globale que vous utilisez. Vous devez utiliser maintenant `$_GET` et non `$_POST`.

Ajouter un champ prénom et vérifier votre URL pour savoir comment injecter plusieurs paramètres à une URL.



Exercice

Créer une page pour la livraison rapide de sandwiches qui s'appelle 'resa.html'.
Ce formulaire devra envoyer les données à une page 'recap.php'.
Ce formulaire devra contenir :

- un champ texte nom
- un champ texte prénom
- Un champ numérique pour le nombre de sandwiches
- Un textarea pour l'adresse
- Une liste déroulante contenant le type du sandwich (viande, poulet, escalope).
- Un checkbox contenant les ingrédients à ajouter (harissa, salade, mayo)

Créer une page PHP 'recap.php' qui affiche une récap de la commande. Le prix total devra aussi être affiché. Le prix d'un sandwich est de 4dt. Si vous dépasser 10 sandwiches, une remise de 10% est appliquée.



Fonctionnalités de base de PHP

- Ne faite jamais confiance aux données envoyées par l'utilisateur. Vérifier les données côté serveur même si vous les avez validé avec du code JS coté client.
- L'une des failles les plus connues est la faille XSS (cross-site scripting). Elle consiste à injecter du code HTML contenant du code JS dans vos pages afin qu'il soit exécuté par les visiteurs su site.
- Pour éviter un tel problème, deux solutions peuvent être utilisées :
 - Echapper le code HTML reçu en utilisant la fonction **[htmlspecialchars](http://php.net/manual/fr/function.htmlspecialchars.php)** qui convertit les caractères spéciaux en entités HTML (<http://php.net/manual/fr/function.htmlspecialchars.php>).
 - Supprimer le code HTML avec **[strip_tags](http://php.net/manual/fr/function.strip-tags.php)** qui supprime les balises HTML et PHP d'un texte (<http://php.net/manual/fr/function.strip-tags.php>).



Exercice

- Tester le code de votre outil de commande de sandwich. Tester la en injectant un code JS qui affiche une alerte disant qu'il n'y a plus de sandwiches.
- Utiliser l'une des fonctions introduites pour éviter ça?
- Est ce que ça fonctionne ?



FILES

- Nous allons maintenant ajouter un champ de type file qui permet d'ajouter une copie de CIN du demandeur de la commande afin d'assurer une certaine crédibilité de la commande.
- Le traitement de ce type de fichier est particulier, en effet on ne transfère plus des types scalaires mais des fichiers qui peuvent être exécutés dans votre serveur.
- On récupère donc du côté du serveur les informations sur cet élément à travers la variable super globale `$_FILES` qui contient les informations sur le fichier.



PHP Accès à la BD

- Pour pouvoir envoyer des fichiers dans un formulaire il faut ajouter **enctype="multipart/form-data"** dans la balise form.
- Afin de récupérer les propriétés du fichier à uploader, on utilise la variable globale `$_FILES` au lieu de `$_POST` ou `$_GET`. Cette variable offre plusieurs informations sur le fichier dont l'information concernant son emplacement.
- Pour copier un fichier dans le serveur on utilise la fonction **copy** qui prend en paramètre le chemin source suivi de la destination.

<https://www.php.net/manual/fr/function.copy.php>



FILES

Supposons que le nom du champ file est fichier, l'accès à la variable `$_FILES['fichier']` retourne un tableau associatif contenant l'ensemble des informations sur le fichier uploadé.

- **Name** : le nom du fichier sur le poste client.
- **Type** : Type MIME du fichier (permet de contrôler le type des fichiers à accepter)
- **Size** : taille en octet du fichier
- **Tmp_name** : nom du fichier temporaire. En effet, le fichier est mis temporairement dans un fichier défini par la directive "upload_tmp_dir" du fichier `php.ini`. Si vous n'enregistrez pas le fichier, il sera perdu.

Si vous voulez enregistrer le fichier dans votre serveur vous pouvez utiliser la fonction

`move_uploaded_file(path_fich_temp, path_fich_src):`

Vous pouvez utiliser `file_exists` qui Vérifie si un fichier ou un dossier existe en cas de besoin.



FILES

Si vous voulez permettre un upload multiple, il vous suffit d'utiliser la même logique que pour les checkbox cad que le nom de votre champ va être suivi de `[]` pour l'informer de stocker les informations dans un tableau.

Remarque : Préparer un dossier pour y mettre vos uploads.



Exercice

Ajouter le champ CIN et gérer son upload à travers le formulaire.

Faite en sorte que le nom du fichier change et soit unique.

Chercher une fonction en PHP qui vous retourne une valeur aléatoire unique.



Fonctionnalités de base de PHP

Les cookies

Lorsque nous voulons conserver des informations tout au long de la navigation d'un utilisateur sur le site, de conserver ses habitudes, des informations utiles ect, on utilise les cookies et les sessions.

- Un cookie est un **fichier texte** enregistré **côté client**.
- Il permet d'enregistrer des informations utiles sur et pour le client qui sont généralement utilisées pour ses prochaines visites
- Pour des raisons de sécurité, les cookies **ne peuvent être lus** que par les pages du **serveur créateur** du cookie en question.
- La **date d'expiration** des cookies est définie par le **serveur** web qui les a créés.
- 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



Fonctionnalités de base de PHP

Exemple d'utilisation

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



Fonctionnalités de base de PHP

Les cookies

Ecrire un cookie

➤ `bool setcookie (string $name, string $value, int $expire = , string $path , string $domain , bool $secure = false , bool $httponly = false)`

- Tout les arguments sauf `$name` ne sont pas obligatoire
- `$name` : nom du cookie
- `$value` : contenu du cookie, n'y stocker pas de mot de passes ou des propriétés critiques ou importantes
- `$expire` : Le temps après lequel le cookie expire. C'est un timestamp Unix, donc, ce sera un nombre de secondes depuis l'époque Unix (1 Janvier 1970). Il faut donc fixer cette valeur à l'aide de la fonction `time()` qui permet d'avoir le timestamp actuel en y ajoutant le nombre de secondes après lequel on veut que le cookie expire. Si le paramètre est omis, le cookie n'est valable que le temps de la connexion du visiteur sur le site
*Exemple : `time()+60*60*24*365` fera expirer le cookie dans 1 an.*
- `Secure` : est une valeur de type boolean qui vaut TRUE si le cookie doit être transmis par une connexion sécurisée (`https://*`) et FALSE dans le cas contraire. C'est la valeur par défaut.

Remarque : Vous pouvez écrire plusieurs valeurs sous un même nom de cookie en utilisant la notation à crochets des tableaux. Cependant le nom ne prend pas de " Exemple : `setcookie("client[pays]");`



Fonctionnalités de base de PHP

Les cookies

Accéder à un cookie

- Pour accéder à un cookie, il faut se souvenir qu'il est stocké dans la variable super globale `$_COOKIES` ce qui fait qu'on lui accède comme n'importe quel variable dans un tableau associatif via son nom.
- Exemple `$_COOKIES['nom']` permet d'accéder au cookie nom.



Fonctionnalités de base de PHP

Supprimer un cookie

- Il n'existe pas une fonction dédiée à la suppression d'un cookie.
- Deux solutions peuvent être utilisées :
 - Renvoyer le cookie grâce à la fonction `setcookie()` en spécifiant uniquement le **nom du cookie à supprimer**.
 - Envoyer un cookie dont la **date d'expiration est passée** en spécifiant par exemple `time()-1`



Fonctionnalités de base de PHP

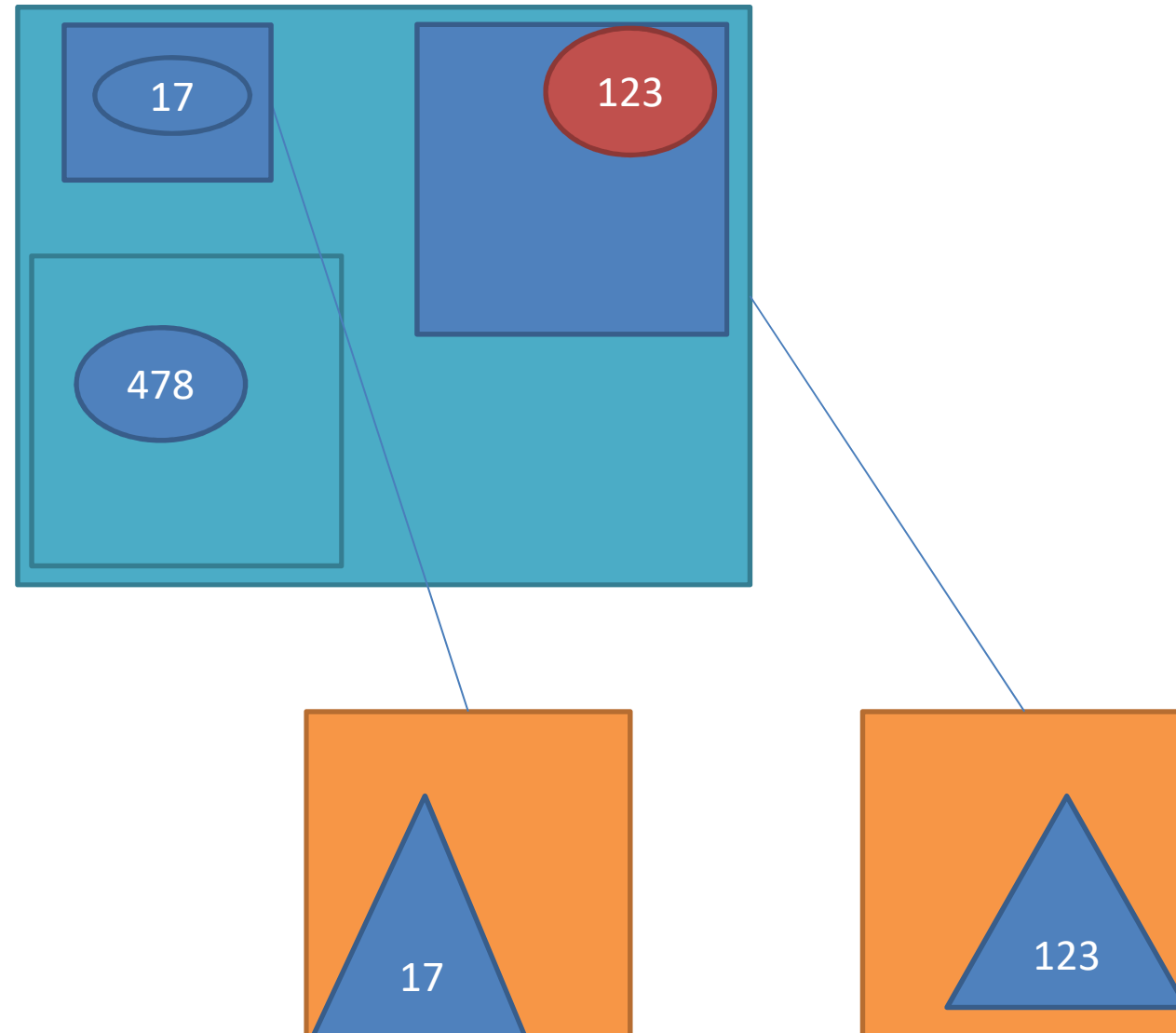
Les sessions sont un mécanisme permettant de mettre en relation les différentes requêtes du **même client** sur une période de **temps donnée**.

- Stocké côté serveur.
- 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 session est identifiée par un numéro d'identification dénommé **identifiant de session (SID)**. Il est transmis soit par un cookie sur le poste client soit ajouté à l'URL.



Fonctionnalités de base de PHP

Les sessions





Fonctionnalités de base de PHP

- Fonctionnement
 - 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.save_path = "c:/wamp/tmp";
```

- Le fichier php.ini peut également préciser la durée de vie d'une session en seconde par **session.gc_maxlifetime**

```
session.gc_maxlifetime = 1800
```



Fonctionnalités de base de PHP

➤ Utilité

- Sécurisé l'accès à vos pages
 - Conserver le user authentifié
 - Restreindre certaines fonctionnalités à des rôles particulier
 - Passer un « token » pour vérifier qu'on suit un même process.
- Conserver des données le long de la visite d'un utilisateur
 - Ces coordonnées
 - Le contenu d'un panier pour un site de e-commerce



Fonctionnalités de base de PHP

- Afin de créer une session on utilise la fonction `session_start()`
- Cette fonction doit être appelée à chaque début de page avant le code HTML.
- Afin de fermer une session on utilise la fonction `session_destroy()`
- Pour ajouter une variable de session on peut :
 - utiliser la méthode `session_register(nomVariable)` (Déconseillé)
 - Ou directement en utilisant la variable superglobale `$_session`
- Pour supprimer les variables de sessions utiliser la fonction `session_unset()`



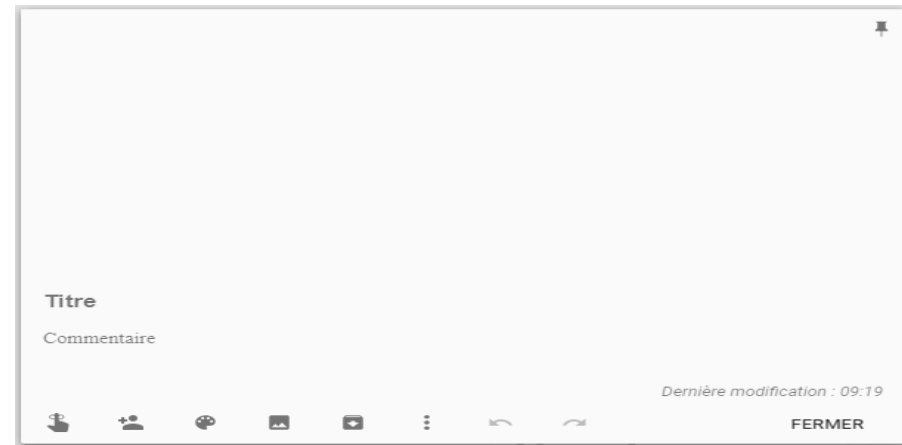
Fonctionnalités de base de PHP

- Afin de nous familiariser avec les sessions, nous allons les utiliser afin de simuler l'outil google keep.

L'idée est d'avoir un formulaire pour ajouter une note.

Nous devons aussi créer une page qui affiche la liste des notes dans la session.

Chaque note doit être affichée dans un bloc qui ressemble à aux notes de google keep :



Fonctionnalités de base de PHP

Les exceptions

- Une exception est une exception à la règle une erreur à gérer.
- En PHP la classe de base des exceptions est la classe **Exception**.
- Une exception peut être lancée (**throw**) et attrapée (**catch**) dans PHP.
- Le code devra être entouré d'un bloc **try** permettant de faciliter la saisie d'une exception potentielle.
- Chaque **try** doit avoir au moins un bloc **catch** ou **finally** correspondant.






Fonctionnalités de base de PHP

Les exceptions

```
<?php
// fonction qui retourne a/b
function quotient($a, $b) {
    if (!$b) {
        throw new Exception('Impossible de diviser par zéro.');
```



```
    }
    return $a/$b;
}
try {
    echo quotient(5,10) . "\n";
    echo quotient(3,0) . "\n";
} catch (Exception $e) {
    echo 'Exception catché : '. $e->getMessage(). "<br>";
}
// Continue execution
echo "Je serais toujours la !<br>";
?>
```



Parmi les méthodes proposées par la Classe Exception nous avons :

`getMessage()` Récupère la chaîne passée au constructeur lors de l'instanciation de l'objet

`getCode()` Récupère le code passé au constructeur.

`getFile()` Récupère le fichier dans lequel a été généré l'exception

`getLine()` Récupère le numéro de la ligne dans lequel a été généré l'exception

`__toString()` Est appelé automatiquement lorsque l'objet de l'exception est utilisé en tant que String. Elle retourne une chaîne décrivant les détails de l'Exception.



Fonctionnalités de base de PHP

- Le block `finally` peut être spécifié après ou au milieu des blocks catch.
- Le code dans le block `finally` sera toujours exécuté après le `try catch` block.
- Le code est exécuté indépendamment du fait qu'une exception soit déclenché ou non.

```
try{  
    if ($test < 0) {  
        throw new Exception('The value have to be Positif');  
    }  
} catch (Exception $e) {  
    echo $e->getMessage();  
} finally {  
    echo "Second finally.\n";  
}
```



Fonctionnalités de base de PHP

➤ Pour les erreurs internes de PHP, PHP utilise la classe `Error`.

Exemple d'erreurs internes:

➤ `ArithmeticError` : Les erreurs de mathématique

➤ `DivisionByZeroError` Déclenché en cas de division par zéro

➤ `TypeError` Déclenché en cas où un argument de type erroné est passé à une fonction ou lorsque la valeur de retour de la fonction est de type erroné ou si le nombre d'arguments passé à la fonction est erroné.

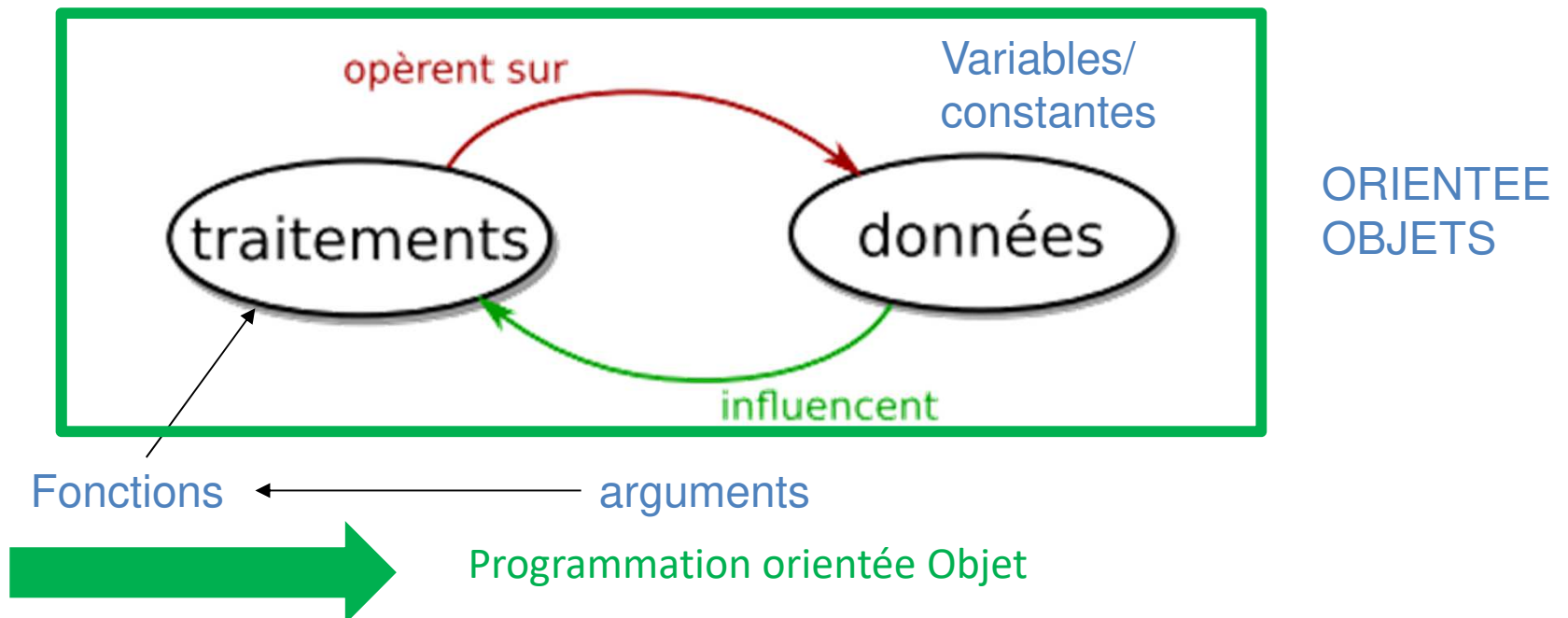


POO PHP5

La programmation
impérative/procédurale (rappel)

- Dans les scripts que vous avez écrits jusqu'à maintenant, nous avons vu les notions
 - de variables/types de données
 - et de traitement de ces données

Séparation entre le
traitement et les données



POO PHP5

Définition d'une classe et d'un objet

- Une classe en orientée objet représente un **regroupement d'attributs** (variables) et de **méthodes** (fonctions).
- Un objet est une instance de classe.



Classe



Objet



POO PHP5

Définition d'une classe et d'un objet

```
class NomClasse
{
    // Contenu de la classe
}
```

Déclaration d'un attribut

```
class NomClasse
{
    VisibilitéDeLaVariable $_nomVariable ;
    // Contenu de la classe
}
```

Exemple :

```
class Gateau
{
    private $_parfum;
}
```




POO PHP5

Définition d'une classe et d'un objet

Déclaration d'une méthode

```
class NomClasse
{
    VisibilitéDeLaVariable $_nomVariable ;
    VisibilitéDeLaFonction function nomDeLaFonction ;
}
```

Exemple :

```
class Gateau
{
    private $_parfum;
    public function appliquerGlacage(){}
    // Contenu de la classe
}
```



POO PHP5

- La visibilité permet de définir de quelle manière un attribut ou une méthode d'un objet est accessible.
- Les 3 niveaux de visibilité en PHP sont :
 - **public** (comportement par défaut) : Accessible partout
 - **private** : Accessible au sein de la classe elle même
 - **Protected** : Accessible au sein de la classe elle-même, ainsi qu'aux classes qui en héritent, et à ses classes parentes.
- Les objets de mêmes types ont accès aux membres privés et protégés les uns des autres.



POO PHP5

Visibilité des attributs et des
méthodes

➤ Concepts de bases :

La POO nous permet de mieux organiser des programmes complexes grâce à quatre notions de bases :

➤ Encapsulation

➤ Abstraction

➤ Héritage

➤ Polymorphisme

Principe d'encapsulation :

regrouper dans le même objet informatique («concept»), les données et les traitements qui lui sont spécifiques :

- **attributs** : les données incluses dans un objet
 - **méthodes** : les fonctions
 - (= traitements) définies dans un objet
- ✓ Les objets sont définis par leurs attributs et leurs méthodes.

Rectangle

Largeur
Hauteur

Surface



POO PHP5

Abstraction

- Pour être véritablement intéressant, une classe doit permettre un certain degré d'abstraction.
- Le processus d'abstraction consiste à identifier pour un ensemble d'éléments :
 - des caractéristiques communes à tous les éléments
 - des mécanismes communs à tous les éléments
- Description **générique** de l'ensemble considéré :
Se focaliser sur l'essentiel, cacher les détails.

Exemple : Rectangles

la notion d'«objet rectangle» n'est intéressante que si l'on peut lui associer des propriétés et/ou mécanismes généraux (valables pour l'ensemble des rectangles)

Les notions de largeur et hauteur sont des propriétés générales des rectangles ([attributs](#)),

Le mécanisme permettant de calculer la surface d'un rectangle ([surface = largeur × hauteur](#)) est commun à tous les rectangles ([méthodes](#))



POO PHP5

Relation entre encapsulation et abstraction

En plus du regroupement des données et des traitements relatifs à une entité, **l'encapsulation** permet en effet de définir **deux niveaux** de perception des objets :

- **niveau externe** : partie « visible » (par les programmeurs-utilisateurs) :
 - **l'interface** : prototypes de quelques méthodes bien choisies
- ✓ résultat du processus d'abstraction
- **niveau interne** : détails d'implémentation
 - **corps** :
 - méthodes et attributs accessibles uniquement depuis l'intérieur de l'objet (ou d'objets similaires)
 - définition de toutes les méthodes de l'objet



POO PHP5

Un petit exemple de la vie réelle

L'interface d'une voiture vis-à-vis du conducteur :

- Volant, accélérateur, pédale de frein, etc.
- Tout ce qu'il faut savoir pour la conduire (**mais pas la réparer ! ni comprendre comment ça marche**)
- L'interface ne change pas, même si l'on change de moteur... et même si on change de voiture (dans une certaine mesure) :
abstraction de la notion de voiture (en tant qu'« objet à conduire »)



POO PHP5

Encapsulation et Interface

Il y a donc deux facettes à l'encapsulation :

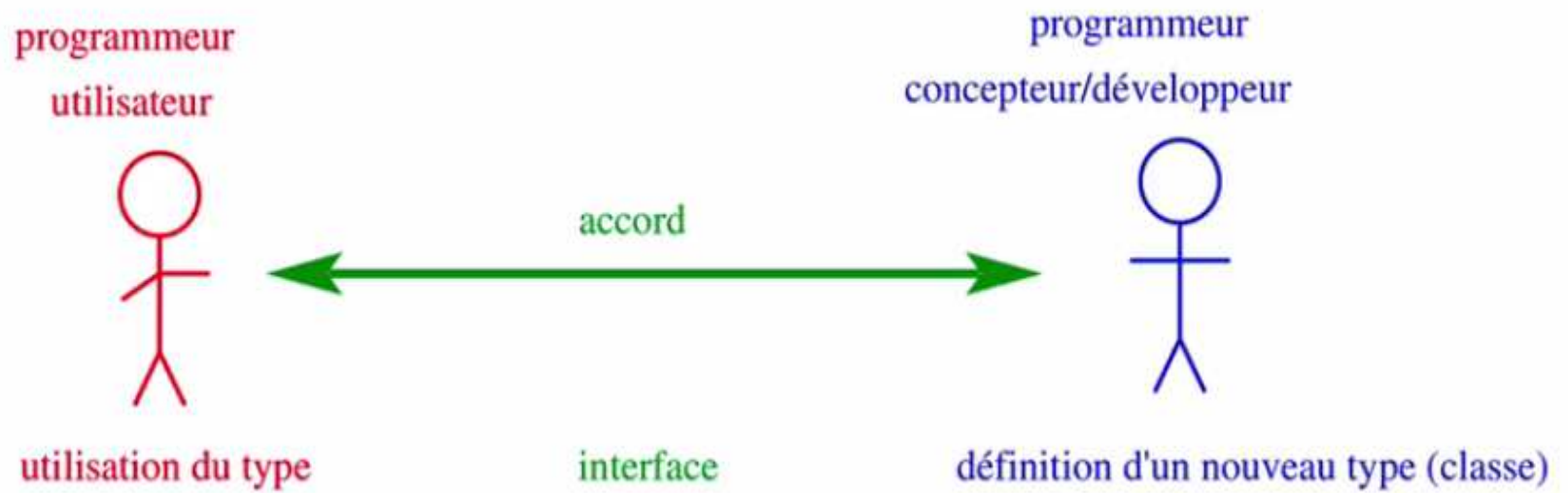
1. regroupement de tout ce qui caractérise l'objet :
données (attributs) et traitements (méthodes)
2. isolement et dissimulation des détails

d'implémentation

Interface = ce que le programmeur-utilisateur (hors de l'objet) peut utiliser

Concentration sur les attributs et les méthodes concernant l'objet (abstraction)

POO PHP5



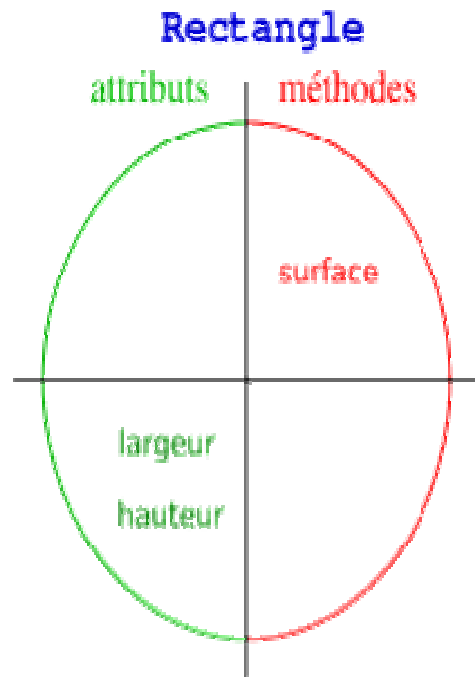


POO PHP5

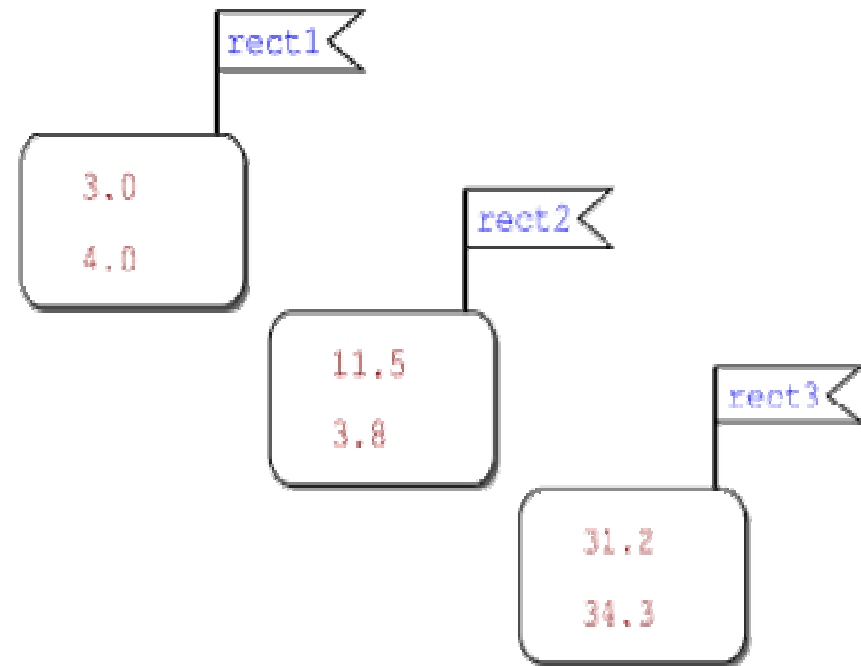
Recap

- L'intérêt de séparer les niveaux interne et externe est de donner un **cadre** plus **rigoureux** à l'utilisation des objets utilisés dans un programme
- Les objets ne peuvent être utilisés qu'au travers de leurs interfaces (les méthodes : niveau externe) et donc les éventuelles **modifications** de la structure interne restent **invisibles** à l'extérieur (même idée que la séparation prototype/définition d'une fonction)
- Une voiture ne peut pas accélérer de 1000 km d'un coup
- Règle : les attributs d'un objet ne doivent pas être accessibles depuis l'extérieur, mais uniquement par des méthodes.

POO PHP5



classe
type (abstraction)
existence conceptuelle
(écriture du programme)



objets/instances
variables en mémoire
existence concrète
(exécution du programme)



POO PHP5

Pour **utiliser** une classe dans un fichier et pouvoir l'instancier, il faut toujours l'appeler en utilisant le mot clé **require**. Il faut indiquer require le chemin à partir du fichier qui utilisera la classe vers la classe. Si les deux fichiers se trouvent dans le même dossier, il suffit d'écrire le nom du fichier.

```
require 'cheminVersMaClasse';
```

Ceci permet de dire à PHP que je vais utiliser cette classe la



POO PHP5

Assez de théorie maintenant qu'on a compris les concepts de bases commençons la création de notre premier objet.

- La création d'un objet à partir d'une classe est **l'instanciation**.
- L'instanciation se fait à l'aide de `new()`;

- Exemple `$monObjet = new MaClasse();`

Pour accéder à un élément de votre objet vous devez utiliser l'opérateur '`->`'

Exemple `$monObjet->maPropriete;`

Afin de référencer l'objet en cours au sein des méthodes de la classe, on utilise la variable `$this`. Cette variable est toujours présente et elle représente l'objet courant.



POO PHP5

Créer un objet

Commençons par créer une classe Voiture qui a les attributs suivants :

- une marque
- une vitesse
- une couleur
- un nombre de chevaux.



POO PHP5

```
class Voiture
{
    private $brand;
    private $speed;
    private $color;
    private $strength;
}
```

Créons maintenant une nouvelle voiture.

```
$ford = new Voiture();
```

Nous avons notre première voiture.

Essaye maintenant d'accéder à ses propriétés ajouter lui une vitesse, une couleur. Tester votre code.

Que se passe t-il ?

En affectant directement une valeur à la vitesse

```
$voiture->vitesse = 1000;
```

vous obtenez le message suivant :

```
Fatal error: Uncaught Error: Cannot access private property Voiture::$strength in C:\xampp\htdocs\gmc\mainClass.php:12 Stack trace: #0 {main} thrown in C:\xampp\htdocs\gmc\mainClass.php on line 12
```

Ce message est normal vu qu'on a mis notre attribut en private.

La solution est d'utiliser les **getters** et les **setters** ou accesseurs et mutateurs.

Les **getters** permettent de **récupérer** les valeur des attributs dont nous voulons partager la valeur. Par convention le nom d'un getter commence toujours par **get** suivi du **nom de l'attribut**. Dans notre cas, c'est **getStrength**.

Les **setters** permettent quant à eux de modifier les valeurs d'un attribut tout en respectant l'encapsulation. En effet votre setter permettra de modifier la valeur en gardant vos règles métiers intactes. Pour le nom c'ets **set** suivi du **nom de l'attribut**. Dans notre cas, c'est **setStrength**.



POO PHP5

Ajoutons donc les méthodes suivantes :

```
public function getStrength()  
{  
    return $this->strength;  
}
```

```
public function setStrength($strength) : void  
{  
    $this->strength = $strength;  
}
```

Réessayer votre code maintenant en utilisant les getters et les setters pour afficher l'attributs après sa modification. Ca marche.



POO PHP5

Dans notre setter et sachant qu'une voiture ne peut dépasser les 200 Km on aurait pu prendre en considération cette règle dans le setter qui deviendrait.

```
public function getStrength()  
{  
    return $this->strength;  
}
```

```
public function setStrength($strength) :  
void  
{  
    if ($strength > 0 && $strength <= 200) {  
        $this->strength = $strength;  
    }  
}
```



POO PHP5

Remarque : New retourne l'identifiant de l'objet crée, donc \$monObjet ne contient pas réellement l'objet mais son **identifiant**.

➤ Que fait alors \$monNouvelObjet=\$monObjet ?
On aura deux variable qui font référence au même objet et non deux objets différents.

Créer une nouvelle variable et affecter à cette variable la voiture que vous avez crée.

Modifier la vitesse de cette variable puis tester si votre première voiture a gardé son ancienne vitesse ou non.



POO PHP5

Cloner un objet

- Afin de **cloner** un objet on utilise le mot clé **clone**.
 - `$monNouvelObjet= clone $monObjet.`
- Ceci permettra d'avoir **deux objets différents** avec les mêmes propriétés.

Tester cette fonctionnalité.

- Afin de parcourir l'ensemble des **propriétés visibles** d'un objet, nous pouvons utiliser le **foreach**.
 - Exemple :

```
foreach($this as $cle => $valeur) {  
    print "$cle => $valeur\n";  
}
```

Cet exemple permet à **chaque itération** d'avoir dans la variable `$cle` le nom de l'attribut et dans la variable `$valeur` sa valeur.

- A chaque instantiation d'un nouvel objet avec `new`, une méthode appelé constructeur est automatiquement appelée.
- Cette méthode s'appelle `__construct()`, elle est implicitement appelé. Comme son nom l'indique, elle permet de construire un objet. Généralement elle permet d'irriguer l'objet en initialisant ces attributs. Elle peut aussi préparer l'objet autrement. Elle peut prendre des paramètres mais ne retourne aucune valeur.
- A la fin du cycle de vie d'un objet, une autre méthode est appelé implicitement, c'est la méthode `__destruct()` qui ne prend pas quant à elle de paramètres et qui permet généralement d'avoir une destruction propre de l'objet. Il servira par exemple fermer un fichier ou une connexion à une base de donnée.

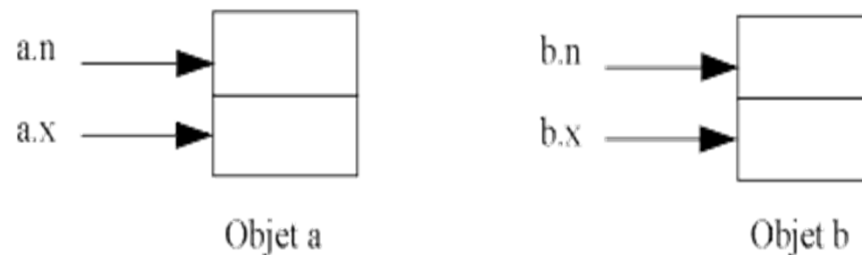


POO PHP5

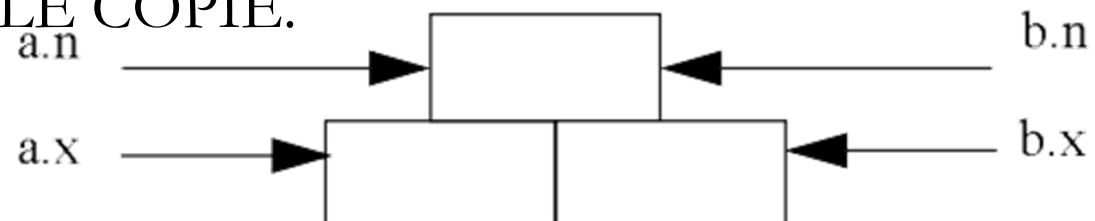
Constructeur et destructeur

```
public function __construct ($brand, $speed,  
$color, $strength)  
{  
    $this->brand = $brand;  
    $this->speed = $speed;  
    $this->color = $color;  
    $this->strength = $strength;  
}  
  
public function __destruct ()  
{  
    echo 'La voiture a été détruite avec  
succées ' ;  
}
```

- Les attributs et méthodes **statiques** sont des **attributs et méthodes de classe**. Elles sont **accessibles via la classe sans** avoir besoin d'instancier un objet.
- Un attribut static est donc un attribut commun à la classe et tous ces instances. Pour les attributs standard si nous déclarons deux objets, chacun aura ses propres attributs.



Par contre si on déclare un attribut n static, il sera alloué une seule fois. Un attribut statique est un attribut partagé par tout le monde. UNE SEULE COPIE.



self

- Les attributs et méthodes statiques sont déclarées avec le mot clé **static**.
- Pour accéder à un attribut ou méthode statique on écrit le nom de la classe suivi de l'opérateur de résolution de portée « :: ».
- **Interdit d'utiliser this !!!!!!!!!!!!!!!!!!!!!!!** qui représente la référence de l'objet.
- « L'équivalent » de this pour la classe est **self**.





POO PHP5

Attributs et méthodes statiques

```
class MaStatic
{
    private $x;
    private static $static1=0;

    /**
     * MaStatic constructor.
     */
    public function __construct()
    {
        self::$static1++;
    }

    public static function nbInstance() {
        echo self::$static1;
    }
}
```

```
$ma1= new MaStatic();
$ma2= new MaStatic();
$ma3= new MaStatic();

MaStatic::nbInstance();
```





POO PHP5

Les constantes de classe

- Les **constantes de classes** sont des attributs constants et qui appartiennent à la classe et non à l'objet.
- Définit avec le mot clé **const**.
- Accessible via l'opérateur de résolution de portée « **::** ».
- Permettent d'éviter des **données muettes**.
- Exemple `const PARIS=«Aéroport Charle de Gaulle»;`

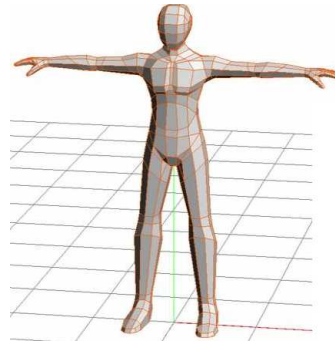


POO PHP5

Les constantes de classe

- Les **constantes de classes** sont des attributs constants et qui appartiennent à la classe et non à l'objet.
- Définit avec le mot clé **const**.
- Accessible via l'opérateur de résolution de portée « **::** ».
- Permettent d'éviter des **données muettes**.
- Par convention, le nommage d'une constante se fait avec des majuscules.
- Ne peuvent contenir que des valeurs primitives (pas d'objet)
- Exemple `const PARIS=«Aéroport Charle de Gaulle»;`

Commençons par un exemple concret : Les personnages de jeux vidéo





POO PHP5

Héritage

Class Guerrier

string nom
int energie
int duree_vie

Arme arme

Rencontrer(Personnage)

Class Voleur

string nom
int energie
int duree_vie

Rencontrer(Personnage)

Voler(Personnage)

Class Magicien

string nom
int energie
int duree_vie

Baguette baguette

Rencontrer(Personnage)

Class Sorcier

string nom
int energie
int duree_vie

Baguette baguette

Baton baton

Rencontrer(Personnage)

PROBLEMES ?

- ✓ Duplication de codes
- ✓ Problèmes de maintenance :
Supposons qu'on veuille changer le nom ou le type d'un attribut, il faudra le faire pour chacune des classes !!!!!

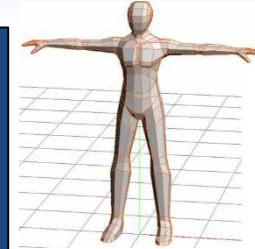
Solution : Héritage

POO PHP5

Class Personnage

Pourquoi ne pas regrouper les caractéristiques en commun dans une super classe ?

```
string nom
int energie
int duree_vie
Rencontrer(Personnage&)
```



Class Voleur

```
Voler(Personnage&)
```



Class Magicien

```
Baguette baguette
```



Class Guerrier

```
Arme arme
```



Class Sorcier

```
Baton baton
```



Héritage

- En PHP pour dire qu'une classe A hérite d'une classe B on utilise le mot clé **extends**.
- La classe fille héritera de toutes les méthodes et les attributs de la classe mère mais elle ne pourra accéder qu'aux attributs **public** et **protected**.
- Une classe fille peut redéfinir les méthodes de la classe mère.
- Pour accéder à une méthode redéfinie de la classe mère on utilise l'objet parent et l'opérateur de portée parent::.





POO PHP5

Héritage

```
class Voiture
{
    private $matricule;
    private $nbChevaux;
    private $couleur;
    private $vitesse;
    public function
__construct($matricule, $nbChevaux,
$couleur, $vitesse)
    {
        $this->matricule = $matricule;
        $this->nbChevaux = $nbChevaux;
        $this->couleur = $couleur;
        $this->vitesse = $vitesse;
    }
    public function tableauDeBord() {
        $date=new DateTime('NOW');
        echo'Bonjour Nous sommes le '.$date-
>format('Y-m-d').' <br>';
        echo'Mon matricule : '.$this-
>getMatricule().' <br>';
        echo'Vitesse actuelle : '.$this-
>getVitesse().' <br>';
    }
    public function getMatricule()
    {
        return $this->matricule;
    }
    public function setMatricule($matricule)
    {
        $this->matricule = $matricule;
    }
}
```

```
public function getNbChevaux()
{
    return $this->nbChevaux;
}
public function
setNbChevaux($nbChevaux)
{
    $this->nbChevaux = $nbChevaux;
}
public function getCouleur()
{
    return $this->couleur;
}
public function setCouleur($couleur)
{
    $this->couleur = $couleur;
}
public function getVitesse()
{
    return $this->vitesse;
}
public function setVitesse($vitesse)
{
    $this->vitesse = $vitesse;
}
public function accelerer(){
    $this->vitesse+=10;
}
public function freiner(){
    $this->vitesse-=10;
}
}
```

Transitivité de l'héritage

- Dans une hiérarchie de classes, la sous-classe hérite de la super-classe :
 - tous les attributs/méthodes (sauf constructeurs et destructeur)
 - le type : on peut affecter un objet de type sous-classe à une variable de type super-classe :



```
Personnage p;  
Guerrier g;  
// ...  
p = g;
```

- L'héritage est transitif : un Sorcier est un Magicien qui est un Personnage

- Afin de forcer la classe à n'être qu'un modèle on la transforme en classe **abstraite**. Une classe abstraite est donc une classe **non instanciable**. Elle ne sert qu'à être une classe mère. On ne peut pas créer une voiture sans marque dans notre exemple.
- Une classe abstraite peut contenir des méthodes « normales » et des **méthodes abstraites**.
- Une **méthode abstraite** est une méthode non définie dans une **classe abstraite**. Elle sert à obliger toute classe héritant de la classe mère de redéfinir cette fonction.
- Une méthode finale est une méthode qui ne peut être redéfinie par une classe fille.





POO PHP5

Héritage

```
Abstract class VoitureAbstraite
{
    private $matricule;
    private $nbChevaux;
    private $couleur;
    protected $vitesse;
    public function __construct($matricule, $nbChevaux, $couleur,
    $vitesse)
    {
        $this->matricule = $matricule;
        $this->nbChevaux = $nbChevaux;
        $this->couleur = $couleur;
        $this->vitesse = $vitesse;
    }
    //Méthode Finale
    final public function quatreFeu() {
        echo "Les quatres feux clignotent !!!";
    }
    //Méthode abstraite
    Abstract public function reglageMoteur();
    public function tableauDeBord() {
        $date=new DateTime('NOW');
        echo'Bonjour Nous sommes le '.$date->format('Y-m-d').' <br>';
        echo'Mon matricule : '.$this->getMatricule().' <br>';
        echo'Vitesse actuelle : '.$this->getVitesse().' <br>';
    }
    ...
}
```

```
class Megane2 extends VoitureAbstraite
{
    public function radarDeRecul()
    {
        echo "Attention Vous allez touchez un obstacle freinez s'il vous
plait";
    }
    // Fonction Statique si on la défini pas il y aura une erreur
    public function reglageMoteur()
    {
        echo " Je règle mon moteur";
    }

    public function tableauDeBord()
    {
        parent::tableauDeBord(); //
        echo "Je suis une Megane";
    }

    public function freiner()
    {
        parent::freiner();
        echo "je dispose de l'abs ";
        if ($this->getVitesse() > 100)
            $this->vitesse -= 30;
    }

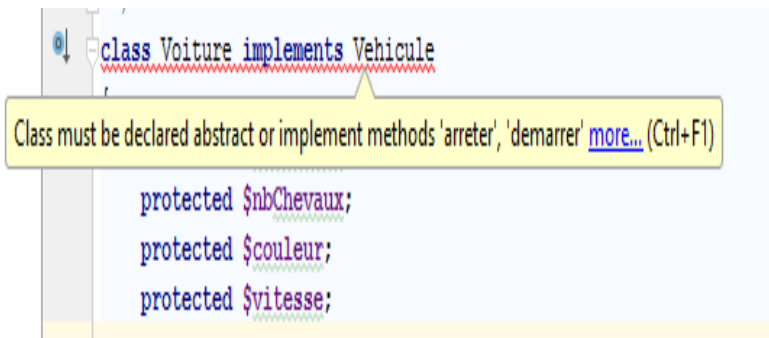
    public function bluetooth()
    {
        echo 'bluetooth Activé votre téléphone est maintenat connecté';
    }
}
```



- Une **interface** est une classe qui permet de spécifier quelles méthodes les classes implémentant cette interface doivent implémenter.
- C'est un **contrat** permettant de préciser exactement le modèle à créer.
- Une interface est un **ensemble de prototype** de méthodes publiques.
- Une interface est déclarée comme une classe. Cependant la place du mot clé class vous devez mettre le **mot clé interface**.
- Afin de spécifier qu'une classe doit implémenter une interface on ajoute devant le nom de la classe le mot clé **implements** suivi du nom de l'interface.
- Une classe implémentant une interface **doit obligatoirement respecter la totalité du contrat. Toutes les méthodes** de l'interface doivent être **implémentées**.



```
interface Vehicule
{
    public function demarrer();
    public function arreter();
}
```



```
class Voiture implements Vehicule
{
    protected $matricule;
    protected $nbChevaux;
    protected $couleur;
    protected $vitesse;

    public function demarrer()
    {
        // TODO: Implement demarrer() method.
    }

    public function arreter()
    {
        // TODO: Implement arreter() method.
    }
}
```





POO PHP5

1. Définissez une classe Vehicule qui a pour attributs des informations valables pour tout type de véhicule : sa marque, sa date d'achat, son prix d'achat et son prix courant. N'oubliez pas de suivre les règles d'encapsulation et d'abstraction.
2. Définissez un constructeur prenant en paramètres la marque, la date d'achat et le prix d'achat.
3. Définissez une méthode publique affiche() qui affiche la valeur des attributs.
4. Sachant qu'un véhicule doit obligatoirement se déplacer et que le déplacement diffère d'un véhicule à l'autre, faites les changements nécessaires pour faire en sorte que toute classe qui hérite de la classe véhicule implémente sa propre fonction de déplacement.
5. Définissez deux classes Voiture et Avion, héritant de la classe Vehicule et ayant les attributs supplémentaires que vous pensez nécessaires.
6. Redéfinissez le constructeur ainsi que la méthode affiche dans les deux classes.
7. Sachant que lors de la vente d'une voiture, 5% du prix de vente doit être versé à l'état, implémentez la méthode taxeVente de la classe voiture qui retourne le montant de la taxe à payer.
8. Instanciez quelques objets et testez vos classes.
9. Vérifiez l'ordre d'exécution des constructeurs et des destructeurs.

Introduction

- Supposons qu'on veuille écrire un code pour un personnage qui va rencontrer des magiciens, des guerrier,... (supposons que l'on ait un tableau de personnages qu'il va rencontrer)
- Que faire sachant que la façon dont un Personnage rencontre un autre peut prendre plusieurs formes : le saluer (Magicien), le frapper (Guerrier), le voler (Voleur)... ??!!!!



Une fonction qui va chercher le type du personnage et appeler sa fonction !!!!!!!!!!!!!!!





POO PHP5

Polymorphisme

Interet

Grâce à l'héritage, le même code pourra être appliqué à un Magicien, un Guerrier , ... qui sont des Personnage.

Si grâce à l'héritage on peut avoir des fonctions différentes selon le personnage pourquoi ne pas avoir un mécanisme qui permet d'avoir une fonction **générique appliquée à tous les personnages** et qui **s'adaptera au type du personnage en ayant un comportement différent, propre à chacun**

POLYMORPHISME

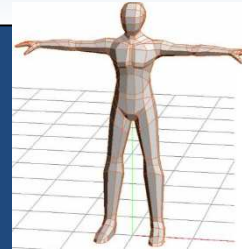
POO PHP5

Polymorphisme

Intérêt
image

Class Personnage

```
string nom
int energie
int duree_vie
Rencontrer(Personnage)
{ saluer() ;}
```



Class Voleur

```
Voler(Personnage)
Rencontrer(Personnage)
{Voler() ;}
```



Class Magicien

```
Baguette baguette
```

Class Sorcier

```
Baton baton
Rencontrer(Personnage)
{Envouter()}
```



Class Guerrier

```
Arme arme
Rencontrer(Personnage)
{frapper()}
```



Etudions le pseudo code d'un joueur qui rencontrera des personnes

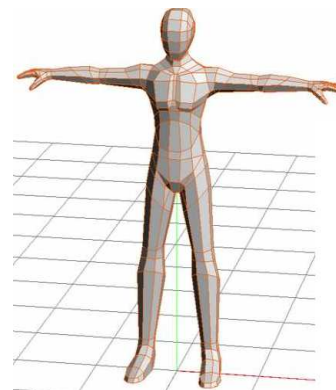
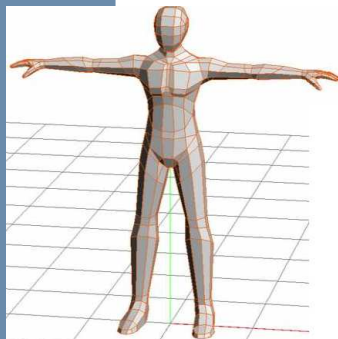
RecontrerPersos()

```
{
    Personnage $joueur;
    $tableauPersonnages = array() ;
    $ tableauPersonnages[] = new Voleur();
    $ tableauPersonnages[] = new Magicien();
```

Si le personnage possède sa propre fonction rencontrer il l'utilise sinon il utilisera la fonction de la super classe

```
foreach ($ tableauPersonnages as $personnage) {
    $joueur.recontrer($personnage);
}
```

Dans la première itération de la boucle c'est un voleur qui est appelé, dans la seconde c'est un magicien



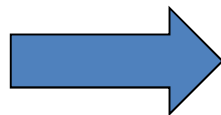
**Pas de
fonction
rencontrer
donc :
Saluer**



Modélisation

Polymorphisme

- Dans une hiérarchie de classes, la sous-classe hérite de la super-classe :
 - tous les attributs/méthodes (sauf constructeurs et destructeur)
 - le type : on peut affecter un objet de type sous-classe à une variable de type super-classe :



```
Personnage p;  
Guerrier g;  
// ...  
p = g;
```

- L'héritage est transitif : un Sorcier est un Magicien qui est un Personnage



POO PHP5

Polymorphisme

Recap

- En POO, le polymorphisme est le fait que les instances d'une sous-classe, lesquelles sont **substituables** aux instances des classes de leur ascendance (en argument d'une méthode, lors d'affectations), gardent leurs **propriétés propres**.
- Le **choix des méthodes à invoquer** se fait lors de l'exécution du programme en fonction de la **nature réelle des instances concernées**.
- La mise en œuvre se fait au travers de :
 - **l'héritage** (hiérarchies de classes) ;



Exercice

1. Créer une classe Produit. Cette classe contient la désignation du produit, son prixHt et sa référence.
2. Créer ensuite la classe Electroménager qui hérite de la classe Produit et qui a comme attributs consommation qui représente la consommation en courant ainsi que TaxeConsommation qui varie selon la valeur de la consommation sur 3 paliers. Si la consommation est inférieure à 100 on a 0% taxe, si la consommation est entre 100 et 300, c'est 10% de taxe, sinon c'est 35% de taxe.
3. Créer finalement la Classe Textile qui a comme attributs taille, couleur et TVA.
4. En utilisant le polymorphisme, nous voulons maintenant et selon les achats dans le panier d'un utilisateur lui afficher le prix Total. La fonction CalculTotalPrix devra être générique dans le sens ou même si on ajoute de nouvelles catégories de Produits cette fonction ne changera pas.



POO PHP5

- Nom de la **classe** DOIT être déclaré comme **StudyCaps**. Le nom de la classe doit être **descriptif**. Il est conseillé **d'éviter les abréviation** tel que Pers pour parler d'une personne.
- Pour les **noms composées** utiliser le séparateur « **_** » et les **majuscules** pour passer d'un nom à un autre ou uniquement les majuscules selon la norme que vous suivez.
- Les **méthodes et les attributs** doivent être nommées en utilisant le **camelCase**, `$parfumGlacage`, `$getParfum()`, ceci s'applique pour la **visibilité non privées**. Par convention les attributs avec une **visibilité privée** doivent avoir un nom commençant par « **_** ». Exemple : `$_parfum`.
- Les **constantes** doivent avoir un nom qui est en totalité en **Majuscule** avec « **_** » en **séparateur**.

- Il faut inclure une Classe afin de l'utiliser un script PHP en utilisant le mot clé **require**.
 - **Exemple : require "ConnexionBD.php"**
- Afin d'éviter de **charger manuellement** chaque classe à utiliser dans un script, il vaut mieux utiliser le concept d'**autoload**.
- PHP possède une **pile d'autoloads** qui permet de lister les fonctions à utiliser pour gérer le **chargement automatique** des **classes instanciées et non déclarées**.
- La solution est donc d'ajouter une fonction qui permet de charger ces classes.
- Afin d'ajouter une fonction à la Pile de fonctions d'auto chargement on utilise la fonction **spl_autoload_register()** qui prend en paramètre une chaîne de caractère représentant le nom de la fonction.





POO PHP5

Auto Chargement

```
function loadClass($maClasse)
{
    require $maClasse . '.php';
}
```





POO PHP5

- Les méthodes magiques sont des méthodes spéciales qui sont appelées implicitement suite à un événement particulier.
- L'idée des méthodes magiques est d'intercepter un événement, de faire un traitement.
- Le nom des méthodes magiques commence par __.
- Nous avons déjà rencontré des méthodes magiques. Qui sont elles ?



- Lorsque vous essayez d'accéder à un attribut innexistant ou private d'un objet, la méthode magique `__get()` est appelé. Elle prend en paramètre le nom de l'attribut appelé.
- Cette fonction retourne une valeur de retour.
- Ajouter cette fonction à une classe et tester la.

```
class MyExampleClass
{
    public function __get ($name)
    {

    }
}
```





POO PHP5

Magic Methods : __get()

```
class MyExampleClass
{
    public function __get ($name)
    {
        echo "You try to access to {$name} attribute, we
        are sorry this attribute is not accessible";
    }
}
```





POO PHP5

Magic Methods : __set()

- De même, lorsque vous accédez à un attribut pour le modifier et qu'il n'existe pas ou que vous n'avez pas le droit de le modifier, la méthode magique `__set()` est invoquée. Cette fonction reçoit deux paramètres qui sont le nom de l'attribut et la valeur à affecter.
- Ajouter cette fonction à une classe et tester la.

```
class MyExampleClass
{
    public function __set($name, $value)
    {

    }
}
```





POO PHP5

Magic Methods : __set()

```
public function __set($name, $value)
{
    echo "you try to change {$name} attribute,
with {$value}we are sorry this attribute is not
accessible";
}
```



- Quand vous essayez d'accéder à une méthode inexistante ou privé pour vous, `__call()` est automatiquement appelée. Cette fonction reçoit deux paramètres contenant le nom de la méthode et en second paramètre un tableau contenant les arguments passés à la méthode.
- Ajouter cette fonction à une classe et tester la.

```
class MyExampleClass
{
    public function __call($name, $arguments)
    {

    }
}
```



- La méthode `__toString()` est quant à elle appelée lorsque vous essayez d'afficher l'objet en tant que string. Par exemple si vous utilisez `echo` pour afficher votre objet.

```
class MyExampleClass
{
    public function __toString()
    {

    }
}
```

- Essayez d'afficher un objet avec `echo`. Que se passe-t-il ?
- Implémentez la méthode `__toString()` afin qu'elle affiche les attributs de votre objet. Retestez encore votre `echo`.





POO PHP5

Magic Methods : __invoke()

- Afin d'utilisez votre **objet en tant que méthode**, vous pouvez utiliser la fonction magique **__invoke()** qui **prend en paramètres autant d'arguments passé lors de l'appel**.

1-Commencez par instancier un objet \$myObject.

2- Utilisez cet objet en tant que fonction comme ceci : \$myObject(1,2,3);

- Que se passe t-il ?

```
class MyExampleClass
{
    public function __invoke($arguments)
    {

    }
}
```

- Implémentez la méthode **__invoke()** (faite en sorte qu'elle affiche juste un message et la liste des paramètres reçues) et réessayer \$myObject(1,2,3);



POO PHP5

Magic Methods

- Official Documentation of Magic Methods :

<http://php.net/manual/en/language.oop5.magic.php>





PHP Accès à la BD

- Pour la gestion des Bases de données, PHP offre plusieurs extensions :
- L'extension **mysql_** : Ensemble de fonction commençons par mysql_ et permettant de communiquer avec mysql. Elle sont devenue **obsolète**.
- L'extension **mysqli_** : ce sont des fonctions améliorées d'accès à MySQL.
- L'extension **PDO** : Outil complet qui permettant une abstraction du type de la base de données traitée permettant ainsi de se connecter aussi bien à MySQL que PostgreSQL ou Oracle.



PHP Accès à la BD

Tableau comparatif

Tableau comparatif

Src : <http://www.php.net/manual/fr/mysqli.overview.php>

	Extension MySQL	Extension mysqli	PDO (avec le pilote PDO MySQL Driver et MySQL Native Driver)
Version d'introduction en PHP	Avant 3.0	5.0	5.0
Inclus en PHP 5.x	Oui, mais désactivé	Oui	Oui
Statut de développement MySQL	Maintenance uniquement	Développement actif	Développement actif depuis PHP 5.3
Recommandée pour les nouveaux projets MySQL	Non	Oui	Oui
L'API supporte les jeux de caractères	Non	Oui	Oui
L'API supporte les commandes préparées	Non	Oui	Oui
L'API supporte les commandes préparées côté client	Non	Non	Oui
L'API supporte les procédures stockées	Non	Oui	Oui
L'API supporte les commandes multiples	Non	Oui	La plupart
L'API supporte toutes les fonctionnalités MySQL 4.1 et plus récent	Non	Oui	La plupart



Nous nous focalisons donc sur **PDO**



PHP Accès à la BD

PDO

- PDO : **P**HP **D**ata **O**bjects
- Extension fournissant des services d'accès aux bases de données.
- Fournie avec plusieurs drivers (MySQL, sqlite, PostgreSQL)
- Disponible par défaut à partir des serveurs PHP 5.1.0



PHP Accès à la BD

- Pour se connecter à une base de données on instancie un objet PDO de la façon suivante :

```
$maDb_connexion = new
```

```
PDO('mysql:host=localhost;dbName=nomDeLaBase', 'userName',  
'motDePasse');
```

On crée donc une nouvelle instance de PDO qu'on récupère dans la variable `$maDb_connexion`. Le constructeur nécessite trois paramètres :

- le driver utilisé, l'adresse du serveur et le nom de la base noté ainsi `driver:host=serveur; dbName=nomDeLaBase'`
- le nom d'utilisateur à utiliser pour la connexion au serveur
- le mot de passe du dit utilisateur

Ces informations diffèrent un peu selon le pilote.



PHP Accès à la BD

- Afin de gérer les erreurs liées à la Connexion à la BD, il faut capturer les erreurs de type **PDOException**.

```
try {  
    $db_connexion = new  
    PDO('mysql:host=localhost;dbname=user1', 'user1',  
    'motdepasse');  
}  
catch (PDOException $e)  
{  
    print "Erreur : " . $e->getMessage();  
    die();  
}
```




PHP Accès à la BD

Pattern Design singleton

- But : Singleton garantit qu'une classe n'a qu'une seule instance et fournit un point d'accès global à cette instance.
- Principe
 - Empêcher les développeur d'utiliser le ou les constructeurs de la classe : déclarer privé tous les constructeurs de la classe.
 - Problème : la classe n'est plus instanciable que par elle-même.
 - Solution : Construire un pseudo constructeur à travers une méthode static. Par convention il sera appelé getInstance.
 - On crée un attribut statique stockant l'unique instance de la classe.
 - Dans getInstance on teste si cet attribut est nul
 - Si null on instancie un objet et on le retourne
 - Sinon on retourne l'instance existante



PHP Accès à la BD

```
<?php
class ConnexionBD
{
    private static $_dbname = "bdphp5";
    private static $_user = "root";
    private static $_pwd = "";
    private static $_host = "localhost";
    private static $_bdd = null;
    private function __construct()
    {
        try {
            self::$_bdd = new PDO("mysql:host=" . self::$_host . ";dbname="
                . self::$_dbname . ";charset=utf8", self::$_user, self::$_pwd,
                array(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES UTF8'));
        } catch (PDOException $e) {
            die('Erreur : ' . $e->getMessage());
        }
    }
    public static function getInstance()
    {
        if (!self::$_bdd) {
            new ConnexionBD();
        } return (self::$_bdd);
    }
}
```



PHP Accès à la BD

Connexion à une BD : afficher
les erreurs

➤ Afin d'afficher les détails des erreurs liées à la BD, il faut activer le suivi de ces erreurs lors de la connexion à la BD.

➤ Exemple :

```
try {  
    $db_connexion = new  
    PDO('mysql:host=localhost;dbname=user1', 'user1', 'motdepasse',  
    array(PDO::ATTR_ERRMODE =>  
    PDO::ERRMODE_EXCEPTION)  
    );  
}  
catch (PDOException $e)  
{  
    print "Erreur : " . $e->getMessage();  
    die();  
}
```



PHP Interroger une BD

Requête simple

- Afin d'interroger une BD via PDO, nous utilisons la méthode `query` qui prend en paramètre la requête à exécuter.

- Exemple :

```
$req=« select * From maTable »;  
$reponse = $_bdd->query($req);
```

- La variable `$reponse` contiendra un objet contenant la réponse de MySQL qui n'est pas directement exploitable.
- Pour exploiter ces données nous utilisons la méthode `fetch` qui retourne une ligne ou `fetchAll` qui retourne un tableau contenant toutes les lignes du jeu d'enregistrements
- L'un des paramètres des méthodes `fetch` et `fetchAll` est l'attribut `fetch_style` qui permet de spécifier le type de la valeur de retour de `fetch` et `fetchAll`



PHP Interroger une BD

Fetch style

- Contrôle comment la prochaine ligne sera retournée à l'appelant. Cette valeur doit être une des constantes `PDO::FETCH_*`.
- `PDO::FETCH_BOTH` (défaut): retourne un tableau indexé par les noms de colonnes et aussi par les numéros de colonnes, commençant à l'index 0, comme retournés dans le jeu de résultats
- `PDO::FETCH_ASSOC` : retourne un tableau indexé par le nom de la colonne comme retourné dans le jeu de résultats
- `PDO::FETCH_NUM` : retourne un tableau indexé par le numéro de la colonne comme elle est retourné dans votre jeu de résultat, commençant à 0
- `PDO::FETCH_OBJ` : retourne un objet anonyme avec les noms de propriétés qui correspondent aux noms des colonnes retournés dans le jeu de résultats
- ...



PHP Interroger une BD

- Exemple de parcours en utilisant des objets :

```
$games = $rep->fetchAll(PDO::FETCH_OBJ);
```

```
foreach($games as $game) :
```

```
    echo $game->nom." - ".$game->commentaires."<br>";
```

```
endforeach;
```

Fetch style



PHP Interroger une BD

Requête paramétrable

- Afin de paramétrer une requête nous pouvons utiliser deux méthodes:
 - Paramétrage manuel en concaténons les paramètres dans la requête. **ENORME FAILLE DE SECURITE SQL INJECTION**
 - les requêtes préparées
- Deux types de requêtes préparées :
 - En utilisant les marqueurs « ? »
 - En utilisant les marqueurs nominatifs



PHP Accès à la BD

- Pour simplifier les choses une requête préparée est une requête dont les paramètres sont insérés dans la fonction lors de l'exécution.
- Elle est effectuée en 2 étapes :
 - Préparer la requête à l'aide de la méthode `prepare`
 - Transmettre les paramètres dans un tableau et exécuter la requête préparée à l'aide de la méthode `execute`
 - Les paramètres sont indiqués dans `l'ordre d'apparition` dans la requête préparée
 - Le contenu des variables est automatiquement sécurisé pour prévenir les risques d'injection SQL.

Exemple : `prepare`

```
$req = $bdd->prepare ('SELECT * FROM personne WHERE nom = ? AND  
age <= ? ORDER BY cin');  
$req->execute(array($nom, $age));
```




PHP Accès à la BD

- Pour simplifier les choses une requête préparée est une requête dont les paramètres sont insérés dans la fonction lors de l'exécution.
- Elle est effectuée en 2 étapes :
 - Préparer la requête à l'aide de la méthode `prepare`
 - Transmettre les paramètres dans un tableau et exécuter la requête préparée à l'aide de la méthode `execute`
 - Les paramètres sont indiqués dans `l'ordre d'apparition` dans la requête préparée
 - Le contenu des variables est automatiquement sécurisé pour prévenir les risques d'injection SQL.

Exemple :

```
$req = $bdd->prepare('SELECT * FROM personne WHERE nom = ? AND  
age <= ? ORDER BY cin');  
$nom=« test »;$age=« 10 »;  
$req->execute(array($nom, $age));
```



PHP Accès à la BD

Requête paramétrable

- Requête paramétrable avec des **marqueurs nominatifs**
- Afin de rendre la requête préparée plus lisible, on peut remplacer les ? Par des **marqueurs nommés**
- Un marqueur nommé est un **nom précédé par « : »**
- **Exemple :**

```
$req = $bdd->prepare('SELECT * FROM personne WHERE  
nom = :nom AND age <= :age ORDER BY cin');  
$req->execute(array('nom'=>$nom, age=>$age));
```

L'ordre des paramètres **n'a plus d'importance** vu que nous utilisons des **tableaux associatifs**.



PHP Accès à la BD

Requête paramétrable

- On peut aussi utiliser la méthode `bindValue` qui prend en paramètres :
 - 1 - rang de l'attribut si on n'utilise pas d'attribut nominatif sinon son nom
 - 2 - Le contenu
 - 3 - Le type (`PARAM_STR`, `PARAM_BOOL`, `PARAM_INT`)
- **`PDO::PARAM_BOOL`** Représente le type de données **booléen**.
- **`PDO::PARAM_NULL`** Représente le type de données **NULL SQL**.
- **`PDO::PARAM_INT`** Représente le type de données **INTEGER SQL**.
- **`PDO::PARAM_STR`** Représente les types de données **CHAR**, **VARCHAR** ou les autres types de données sous forme de chaîne de caractères SQL.
- Exemple :

```
$req = $bdd->prepare('SELECT * FROM personne WHERE nom = ? AND age <= ? ORDER BY cin');  
$req->bindValue(1,"Aymen",PDO::PARAM_STR);  
$req->bindValue(2,20,PDO::PARAM_INT);  
$req->execute();
```



PHP Accès à la BD

- Afin de récupérer le nombre d'enregistrement retourné par la requête on utilise la méthode `rowCount`.

Exemple :

```
$req="select * maTable";
```

```
$rep = $bdd->query($req);
```

```
echo "le nombre d'enregistrements est :".$rep->rowCount();
```

- Afin de récupérer l'id du dernier enregistrement, on utilise la méthode `lastInsertId`.

Remarque : Ca ne marche qu'après un INSERT.

Exemple :

```
echo "le dernier id est :".$bdd->lastInsertId()."<br>";
```

PHP Accès à la BD

- Afin d'ajouter, modifier et supprimer un enregistrement dans la Base de données PDO nous offre la méthode **exec**.
- Cette méthode prend en paramètre la requête à exécuter.
- On peut utiliser la méthode **prepare** afin de préparer la requête à exécuter.
 - `$req= $bdd->prepare(« La requête à préparer »)`
- Une fois la requête préparée, on utilise la méthode **execute** en lui passant un tableau associatif contenant la liste des paramètres.

```
$req= $bdd->prepare("insert into matable
(`champ1`, `champ2`, `champn`) VALUES
(:val1, :val2, :valn)");

$req->execute(array(
    'val1'=>'val1',
    'val2'=>'val2',
    'valn'=>5
));
```





PHP Accès à la BD

Modification d'enregistrement

- Même fonctionnement que l'ajout.
- Requête update.

```
$req= $bdd->prepare (« update matable set  
champ1=:val1, champ2= :val2, champ3=  
:champ3 where champ_condition= :cnd" );  
  
$req->execute (array (  
    'val1'=> 'newval1',  
    'val2'=> 'newval2',  
    'valn'=> 7,  
    'cnd'=> 'valCnd',  
));
```



PHP Accès à la BD

Suppression d'enregistrement

- Même fonctionnement que la suppression
- Requête delete.

```
$req= $bdd->prepare (« delete from matable  
where champ_condition= :cnd" );  
  
$req->execute (array (  
    'cnd'=>'valCnd',  
));
```

