

</>

Cours de PHP

May the force be with you !

INTRODUCTION

PHP: **P**re **H**ypertext Processor, plus connu sous son sigle PHP (sigle auto-référentiel), est un langage de programmation libre.

Il est considéré comme une des bases de la création de sites web dits dynamiques mais également des applications web.

Date de la première version : **1994**



Rasmus Lerdorf

LES BASES DE PHP

Les variables

Les tableaux

Les boucles

Les conditions

Les opérateurs logiques

Exemple de XSS

Gestion des exceptions

Générer un nombre aléatoire

Générer un nombre aléatoire sécurisé

Les fonctions anonymes

Timestamp & La gestion des dates

La connexion à une base de données

LES BASES DE PHP

La PHPDoc
Les constantes magiques

Les namespaces
La fonction Levenshtein
Les fonctions récursives
Les expressions régulières

Cas pratique: "Création d'un Scrapper"

La limite mémoire de PHP
Round, Floor & Ceil
Les traits

Exercice

Pour les nombres entre 20 et 60 :

- si le nombre est divisible par 3 : on écrit "Game"
- si le nombre est divisible par 5 : on écrit "Over"
- si le nombre est divisible par 3 et par 5 : on écrit "Game Over"
- sinon : on écrit le nombre



Solution

```
for ($i = 20; $i < 61; $i++) {  
  
    if ($i % 3 === 0 && $i % 5 === 0) {  
        echo '<p>GAME OVER</p>';  
    } elseif ($i % 3 === 0) {  
        echo('<p>GAME</p>');  
    } elseif ($i % 5 === 0) {  
        echo '<p>OVER</p>';  
    } else {  
        echo '<p>' . $i . '</p>';  
    }  
  
}
```

```
for ($i = 20; $i < 61; $i++) {  
  
    $Game = $i % 3 === 0 ? 'Game' : '';  
    $Over = $i % 5 === 0? 'Over' : '';  
    echo '<p>' . (!empty($Game.$Over) ? $Game.$Over:$i) . '</p>'  
}
```

Les variables

Qu'est-ce que c'est ?

Une variable en PHP est **un espace de stockage dans la mémoire** qui permet de conserver une valeur ou un ensemble de valeurs. Cette valeur peut être de différents types, comme un nombre, une chaîne de caractères, un tableau, un objet, etc.

Les variables en PHP sont dynamiques, ce qui signifie que leur type est déterminé automatiquement en fonction de la valeur qui leur est assignée, et elles peuvent changer de type à tout moment.

Comment nommer une variable ?

Les noms de variables en PHP doivent commencer par un signe dollar (\$), suivi d'une lettre ou d'un underscore (_).

Les caractères suivants peuvent être des lettres, des chiffres ou des underscores. Les noms de variables ne doivent pas contenir d'espaces ni de caractères spéciaux, autres que l'underscore.

Exemple de nommage:

\$name	✓
\$_price	✓
\$is-member	✗
\$2dates	✗
\$question?	✗
\$FirstName	✓
\$total_price	✓

Les variables

Quelle est la différence entre "" & ''

```
$username = "Alice";  
  
echo "Bonjour $username";
```

VS

```
$username = 'Alice';  
  
echo 'Bonjour '.$username;
```

Les bonnes pratiques

Utilisez des noms de variables clairs et significatifs, reflétant leur contenu ou leur rôle dans le code. Par exemple, pour une variable contenant un tableau, on mettra le nom de la variable au pluriel.

Utilisez une convention de nommage uniforme. En PHP, les noms de variables sont généralement écrits en camelCase.



La portée des variables

Les variables globales

Les variables définies en dehors de toute fonction ou méthode ont une portée globale. Elles sont accessibles depuis n'importe quel point du script, sauf à l'intérieur des fonctions, à moins d'utiliser une astuce.

```
$variableGlobale = "Je suis globale";  
  
1 usage  
function maFonction() {  
    // Bah oui, ça marche pas  
    echo $variableGlobale;  
}  
  
maFonction();
```

```
$variableGlobale = "Je suis globale";  
  
1 usage  
function maFonction() {  
    // On importe la variable globale  
    global $variableGlobale;  
    // Forcément, ça marche mieux  
    echo $variableGlobale;  
}  
  
maFonction();
```

La portée des variables

```
3 usages

function compteur() {
    // La variable n'est pas réinitialisée
    static $count = 0;
    $count++;
    echo $count;
}

compteur(); // Affiche : 1
compteur(); // Affiche : 2
compteur(); // Affiche : 3
```

Les variables static

Les variables définies avec le mot-clé static dans une fonction ont une portée statique. Une variable statique conserve sa valeur entre plusieurs appels de la fonction, contrairement aux variables locales classiques qui sont réinitialisées à chaque appel de la fonction.

Les tableaux

Indexé

Un tableau indexé est une structure de données qui permet de stocker plusieurs valeurs sous un même nom de variable. Chaque élément du tableau est accessible à l'aide d'un index numérique qui représente sa position dans le tableau. L'index commence généralement à 0 pour le premier élément, 1 pour le deuxième, et ainsi de suite.

```
$array = [  
    "Premier élément",  
    "Deuxième élément",  
    "Troisième élément"  
];  
  
$array[0] = "Premier élément";
```

Associatif

Contrairement aux tableaux indexés où les éléments sont accessibles par un index numérique, les éléments d'un tableau associatif sont accessibles à l'aide de clés qui peuvent être des chaînes de caractères ou des entiers.

```
$associativeArray = array(  
    "nom" => "Transcène",  
    "prénom" => "Jean",  
    "âge" => 30,  
    "ville" => "New York"  
);  
  
$associativeArray["nom"] = "Transcène";
```

Les superglobales

Qu'est ce que c'est ?

Les superglobales en PHP sont des variables prédéfinies accessibles de n'importe où dans le script, que ce soit au niveau global, dans des fonctions, ou dans des méthodes. Elles permettent de gérer les données d'entrée utilisateur, les variables d'environnement, les cookies, les sessions, etc.

Les superglobales sont des tableaux associatifs qui contiennent des informations importantes sur les requêtes HTTP, l'environnement serveur, les sessions et bien plus encore. Elles sont toujours disponibles sans avoir besoin de déclarations spéciales.

<code>\$_GET</code>	Contient les données envoyées via la méthode GET dans une requête HTTP (URL).
<code>\$_POST</code>	Contient les données envoyées via la méthode POST dans une requête HTTP.
<code>\$_REQUEST</code>	Contient les données issues des requêtes GET, POST, et des cookies.
<code>\$_FILES</code>	Contient les informations sur les fichiers téléchargés via un formulaire.
<code>\$_SESSION</code>	Contient les informations de session des utilisateurs (persistantes entre différentes pages).
<code>\$_COOKIE</code>	Contient les données des cookies envoyées par le client dans une requête HTTP.
<code>\$_SERVER</code>	Contient les informations sur le serveur et l'environnement d'exécution.
<code>\$_ENV</code>	Contient les variables d'environnement (variables système).
<code>\$_GLOBALS</code>	Un tableau contenant toutes les variables globales définies dans le script.

Les variables de session

```
session_start();
$_SESSION['value'] = "Ma valeur";
unset($_SESSION['value']);
```

La variable de sessions est un tableau associatif des valeurs stockées dans les sessions, et accessible au script courant.

C'est-à-dire que cette variable est accessible depuis une autre page PHP.

On déclare une variable de session comme ci-contre:

Les tableaux

Multi-dimensionnel

Cela signifie qu'au lieu d'avoir des valeurs simples comme des nombres ou des chaînes de caractères, les éléments du tableau sont eux-mêmes des tableaux.

```
$arrays = array(  
    array(1, 2, 3),  
    array(4, 5, 6),  
    array(7, 8, 9)  
,
```

Les fonctions de tri

- **sort(\$array)**: trie dans l'ordre alphanumérique les éléments du tableau.
- **rsort(\$array)**: trie dans l'ordre alphanumérique inverse les éléments du tableau.
- **asort(\$array)** : trie un tableau associatif en ordre croissant en préservant les associations clé-valeur.
- **arsort(\$array)** : trie un tableau associatif en ordre croissant en préservant les associations clé-valeur.
- **ksort(\$array)** : trie un tableau associatif en ordre croissant en fonction des clés.
- **krsort(\$array)** : trie un tableau associatif en ordre décroissant en fonction des clés.
- **usort(\$array, \$function)** : trie un tableau en utilisant une fonction de comparaison définie par l'utilisateur.
- **uasort(\$array, \$function)** : trie un tableau associatif en utilisant une fonction de comparaison définie par l'utilisateur.

La fonction “array_map”

La fonction “array_map” prend pour paramètre une fonction qui va s’appliquer à chaque élément du tableau passé en deuxième paramètres.

Le résultat de cette fonction est un tableau contenant le résultat de la fonction appelé pour tous les éléments du tableau.

```
function cube($n)
{
    return ($n * $n * $n);
}

$array = [1, 2, 3, 4, 5];
$result = array_map('cube', $array);

var_dump($result, $array);
```

```
$func = function($value) {
    return $value * $value * $value;
};

$result = array_map($func, $array);
```

```
$result = array_map(fn ($value) => $value * $value * $value, $array);
```

La fonction “array_filter”

Cette fonction prend deux paramètres: le tableau à filtrer (\$array), et une fonction qui définit la condition du filtre (\$callback)

La fonction de rappel callback évalue chaque valeur du tableau passé en paramètre.

Si la fonction de rappel callback retourne true, la valeur courante du tableau est retournée dans le tableau résultant.

```
function modulo3($element){  
    return $element % 3 === 0;  
}  
  
$result = array_filter(range( start: 20, end: 60), callback: 'modulo3');
```

```
$func = function ($element){  
    return $element % 3 === 0;  
};  
  
$result = array_filter(range( start: 20, end: 60), $func);
```

```
$result = array_filter(range( start: 20, end: 60), fn($element) => $element % 3 === 0);
```

La fonction “array_reduce”

Un cas d'utilisation courant de array_reduce est le calcul de la somme, la concaténation de chaînes, ou toute autre opération qui nécessite une réduction d'un ensemble de valeurs en une seule valeur agrégée.

Cette fonction prend deux paramètres: le tableau à réduire(\$array), et une fonction qui définit comment réduire (\$callback), et la valeur de départ (\$initial).

```
$numbers = [1, 2, 3, 4, 5];

$total = array_reduce($numbers, function($carry, $item) {
    return $carry + $item;
}, 10);

echo $total; // Cela affichera 25
```

La fonction “array_keys”

La fonction array_keys() en PHP est utilisée pour retourner toutes les clés présentes dans un tableau. Elle peut aussi être utilisée pour récupérer uniquement les clés correspondant à une certaine valeur si un deuxième argument optionnel est passé.

```
$table = [  
    'a' => 1,  
    'b' => 2,  
    'c' => 3,  
];  
  
$keys = array_keys($table);  
// Affiche : ['a', 'b', 'c']  
var_dump($keys);
```

```
$table = [  
    'a' => 1,  
    'b' => 2,  
    'c' => 1,  
];  
  
$keys = array_keys($table, filter_value: 1);  
// Affiche ['a', 'c']  
var_dump($keys);
```

Les tableaux

Quelques fonctions utiles pour les tableaux:

- **count(\$tab)** ou **sizeof(\$tab)**: retourne un entier qui indique le nombre d'entrées du tableau.
- **in_array(\$var,\$tab)**: vérifie si la variable \$var existe dans le tableau. Si oui la fonction in_array() retourne true sinon elle retourne false.
- **list(\$var1,\$var2,\$var3...)**: affecte chacune des entrées du tableau respectivement au variables \$var1, \$var2, \$var3...
- **shuffle(\$tab)**: mélange le contenu du tableau en changeant l'index des entrées aléatoirement.
- **array_merge(\$tab1,\$tab2,\$tab3...)**: retourne un seul grand tableau qui contient les éléments des tableaux \$tab1, \$tab2, \$tab3...
- **implode(\$sep,\$tab)** ou **join(\$sep,\$tab)**: retourne une chaîne de caractères constituée des éléments du tableau séparés par le contenu de la variable \$sep.
- **explode(\$occ,\$str)**: cette fonction s'applique sur les chaînes de caractères. Elle crée un tableau en éclatant la chaîne \$str au niveau des occurrences \$occ.

Les boucles

Boucle "For"

```
for ($i = 1; $i < 21; $i++){  
    echo $i;  
}
```

Boucle "Foreach"

```
foreach (range( start: 1, end: 5) as $item){  
    echo $item;  
}
```

Boucle "Do...While"

```
$i = 0;  
do {  
    echo $i;  
    $i++;  
} while ($i < 5);
```

Boucle "While"

```
$i = 0;  
  
while($i <= 10){  
    echo '$i contient la valeur ' . $i. '<br>';  
    $i++;  
}
```

Les boucles avec références



Erreur courante avec "Foreach" en PHP

Ce type de boucle ne vous permet pas modifier les éléments que vous parcourez. Ici "\$fruit" est une copie de l'élément du tableau par conséquent, vous pouvez le modifier localement mais **vos modifications n'affectent pas les éléments du tableau.**

Afin de pouvoir modifier les éléments du tableau avec ce type de boucle, vous devez utiliser une référence pour pointez vers l'élément original du tableau.

```
$fruits = array("pomme", "orange", "banane", "fraise");

foreach ($fruits as $fruit) {
    $fruit = "jus de " . $fruit;
}
```

```
$fruits = array("pomme", "orange", "banane", "fraise");

foreach ($fruits as &$fruit) {
    $fruit = "jus de " . $fruit;
}
```

Les conditions

Condition Ternaire

L'opérateur ternaire permet de faire une évaluation conditionnelle en une seule ligne.

```
$result = 1 === 2 ? 'OK' : 'Pas OK';
```

Condition "if ... elseif ... else"

```
$boolean = true;

if($boolean === true){
    echo 'La valeur est "True"';
} elseif ($boolean === false){
    echo 'La valeur est "False"';
} else {
    echo 'La valeur n\'est pas un boolean';
}
```

Condition "switch case"

```
switch($fruit) {
    case "pomme":
        echo "Les pommes sont rouges.";
        break;
    case "banane":
        echo "Les bananes sont jaunes.";
        break;
    default:
        echo "Je ne connais pas cette fruit.";
}
```

Condition "match" (**PHP 8 only**)

```
$note = 4;

$evaluation = match($note) {
    1 => 'Médiocre',
    2, 3, 4, 5, 6, 7, 8, 9 => 'Peut mieux faire',
    10, 11, 12 => 'Passable',
    13, 14, 15, 16 => 'Bien',
    17, 18, 19, 20 => 'Très bien !!!',
    default => 'Non Évalué'
};
```

Les opérateurs logiques

```
$result = ($a and $b); // true si $a ET $b valent true.  
$result = ($a or $b); // true si $a OU $b valent true.  
$result = ($a xor $b); // true si $a OU $b est true, mais pas les deux en même temps.  
$result = !$a; // true si $a n'est pas true.  
$result = ($a && $b); // true si $a n'est pas true.  
$result = ($a || $b); // true si $a OU $b est true.
```

```
$a = (false && foo());  
$b = (true || foo());  
$c = (false and foo());  
$d = (true or foo());
```



Dans l'exemple ci-contre, la fonction 'foo()' n'est jamais appelée.

La priorité des opérateurs logiques



!	Différent de
&&	ET
	OU
=	Affectation
AND	ET
XOR	OU Exclusif
OR	OU

```
// Retourne false  
$result = true && false
```

VS

```
// Retourne true  
$result = true AND false
```

Gestion des erreurs

La gestion des erreurs est une partie essentielle du développement en PHP. Elle permet de gérer les situations inattendues et de garantir une expérience utilisateur fluide.

Pour gérer ces exceptions en PHP, on utilise l'instruction "try...catch...finally", cela permet de contrôler le comportement du programme en cas d'erreur, plutôt que de laisser le script s'arrêter brutalement.

Exemple :

```
try {
    $fichier = fopen("fichier_inexistant.txt", "r");
} catch (Exception $e) {
    echo "Une erreur s'est produite : " . $e->getMessage();
} finally {
    if (isset($fichier)) {
        fclose($fichier);
    }
}
```

La fonction htmlspecialchars()

Qu'est ce que c'est et pourquoi l'utiliser ?

La fonction **htmlspecialchars()** en PHP est utilisée pour échapper les caractères spéciaux HTML dans une chaîne de caractères. Cela signifie qu'elle convertit certains caractères spéciaux en leurs entités HTML correspondantes, ce qui est essentiel pour prévenir les failles de sécurité comme les injections de code HTML ou de scripts (**XSS**).

Exemple :

```
// Valeur soumise par l'utilisateur
$name = $_POST['name'];
// Si l'utilisateur entre <script>alert('Hacked!');</script>
echo "Bienvenue, $name!";
```



Cela entraînera l'exécution du script JavaScript et une alerte apparaîtra, ce qui représente une faille de sécurité (XSS).

```
// Valeur soumise par l'utilisateur
$name = htmlspecialchars($_POST['name']);
// Si l'utilisateur entre <script>alert('Hacked!');</script>
echo "Bienvenue, $name!";
```

Cela affichera :

Bienvenue, <script&gtalert('Hacked!');</script&gt

BLAKE ET MORTIMER



Exercice

Énoncé:

Vous allez créer une application web simple pour gérer une collection d'albums de bande dessinée du célèbre duo "Blake et Mortimer". Vous disposerez d'une liste prédefinie de types de bandes dessinées ainsi que des détails sur chaque album.

Tâches à réaliser :

- Créer un fichier index.php qui servira de page d'accueil.
- Intégrer Bootstrap ou Tailwind pour un affichage attrayant et responsive sous forme de "card".
- Afficher les albums de bande dessinée avec leurs détails et leur type.
- Faire un champ de recherche pour être capable de trouver un tome par l'une des informations suivantes: Nom, Scénariste, Dessinateur
- Faire une fonction pour afficher le nombre d'albums publié entre 2000 & 2010
- Afficher dans un tableau la liste des dessinateurs et le nombre d'album qu'ils ont fait.
- Afficher dans un tableau la liste des Scénariste et le nombre d'album qu'ils ont fait

Lien vers l'exercice : <https://codeshare.io/GApwD8>

La limite mémoire PHP

Le limite mémoire de PHP est par script, tout comme une limite de mémoire standard suffisante pour toute application Web. La limite par défaut est de 128M.

Si les scripts PHP tentent d'utiliser plus de 128 Mo, ces scripts renverront actuellement des erreurs de dépassement de limite de mémoire.

```
$size = memory_get_usage();  
  
$unit= ['b','kb','mb','gb','tb','pb'];  
$memoryUsed = @round( num: $size/pow( num: 1024, ($i=floor(log($size, base: 1024)))), precision: 2). . '$unit[$i];
```

Round , Floor & Ceil

Les fonctions ci-dessous permettent d'arrondir une valeur décimale.

Pour la fonction 'round', on peut y passer un deuxième paramètre afin de définir le nombre de chiffres après la virgule.

```
$float = 4.7;  
  
$ceiled = ceil($float);      // 5  
$floored = floor($float);    // 4  
$rounded = round($float);    // 5
```

Les Traits

Un trait est semblable à une classe, mais il ne sert qu'à grouper des fonctionnalités d'une manière intéressante. Il n'est pas possible d'instancier un Trait en lui-même.

```
trait Inventaire
{
    public $nombre;

    public function plusUn()
    {
        $this->nombre++;
        echo $this->nombre . '<br>';
        return $this;
    }
}
```

Les traits permettent d'éviter une redondance de code dans les classes.

On l'appelle comme ci-dessous.

```
use Inventaire;
```

Générer un nombre aléatoire

Les fonction **rand()** et **mt_rand()**



Les fonctions comme rand() et mt_rand() utilisent des générateurs de nombres pseudo-aléatoires (PRNG). Un **PRNG** est un algorithme qui produit une séquence de nombres qui ont les propriétés statistiques des nombres aléatoires. Cependant, cette séquence est entièrement déterminée par une valeur initiale appelée graine ou seed.

mt_rand() est plus avancé que rand() grâce à l'utilisation de l'algorithme **Mersenne Twister**.

En PHP, vous pouvez d'ailleurs initialiser la seed de ses fonction avec la méthode **srand()** et **mt_srand()** !

Générer un nombre aléatoire sécurisé



Pourquoi random_int() ? plutôt que rand()

C'est un générateurs de nombres aléatoires sécurisés cryptographiquement (**CSPRNG**).

Random_int() est conçu pour être sécurisé du point de vue cryptographique, ce qui le rend idéal pour générer des clés d'authentification, des mots de passe temporaires, etc.

Un CSPRNG tire sa "**seed**" de sources d'entropie qui sont imprévisibles, comme le mouvement de la souris, les timings des frappes au clavier, le bruit de fond matériel ou des instructions spécifiques au processeur. Ces sources fournissent un niveau de hasard difficile à prévoir même pour un attaquant.

Exercice

Vous allez devoir créer votre propre méthode pour générer un nombre pseudo-aléatoire, en utilisant des concepts mathématiques de base et des sources d'entropie disponibles en PHP.

Votre générateur devra être capable de produire un entier aléatoire entre deux bornes définies (par exemple, entre 1 et 100).



Les fonctions anonymes

Définition

Les fonctions anonymes, également connues sous le nom de "**closures**" en PHP, sont des fonctions qui n'ont pas de nom associé. Elles sont très utiles pour des tâches ponctuelles et peuvent être assignées à des variables ou utilisées comme arguments de fonction.

Exemple:

```
$somme = function($a, $b) {
    return $a + $b;
};

$resultat = $somme(10, 20);
```

```
1 usage
function operation($x, $y, $fonction) {
    return $fonction($x, $y);
}

$resultat = operation(10, 20, function($a, $b) {
    return $a - $b;
});
```

Passer une référence dans une fonction

Fonctionnement

Par défaut, les arguments d'une fonction sont passés par valeur, ce qui signifie que toute modification apportée à l'intérieur de la fonction n'affecte pas la variable d'origine. Cependant, il est possible de passer un paramètre par référence (en utilisant le symbole &), ce qui permet à la fonction de modifier directement la variable originale.

Exemple:

```
1 usage
function ajouterCinq(&$nombre) {
    $nombre += 5;
}

$valeur = 10;
ajouterCinq( &nombre: $valeur);
echo $valeur; // égale à 15
```

Qu'est-ce qu'un Timestamp UNIX ?

Définition

Un timestamp UNIX est un entier qui représente le nombre de secondes écoulées depuis le 1er janvier 1970 à 00:00:00 UTC. Ce nombre peut être positif (pour les dates futures) ou négatif (pour les dates avant 1970).

Le timestamp est souvent stocké en tant qu'entier de 32 bits, bien que des versions 64 bits soient de plus en plus utilisées pour étendre la plage de dates possible.

Pourquoi ?

Ce format a été choisi pour sa simplicité et son uniformité pour représenter le temps. Il permet aux systèmes d'exploitation et aux applications de calculer et comparer les dates et heures de manière uniforme, sans avoir à gérer les complications des fuseaux horaires ou des formats de dates différents.

Origine du Timestamp UNIX



Le 1 Janvier 1970 à minuit UTC

Même si le temps est une notion vieille comme le monde, dans l'informatique le temps à un point de départ.

Ce point de départ qu'on pourrait assimiler au Big Bang de l'informatique s'appelle l'époque UNIX et commence le 1 Janvier 1970 à minuit UTC.

Ce point de départ a été inventé au début des années 70 par Dennis Ritchie & Ken Thompson qui sont également les créateurs du langage C et du premier système d'exploitation UNIX.



Ken Thompson



Dennis Ritchie

Le Bug de l'an 2038



Le plus grand risque associé aux timestamps UNIX est le "Bug de l'an 2038".

Ce problème survient parce que dans les systèmes utilisant des entiers de 32 bits pour stocker les timestamps, le nombre maximal de secondes qu'ils peuvent représenter est compris entre -2 147 483 648 et 2 147 483 647 ce qui correspond au 19 janvier 2038 à 03:14:07 UTC. Cela représente une plage d'environ 136 ans.

Pour résoudre ce problème temporairement sur un système 32 bit, il faut passer les entiers signés en entiers non signés ce qui permet de repousser le bug jusqu'en 2106.

La solution la plus viable pour résoudre ce problème serait de tout migrer en 64 bit

Datetime Vs DateTimeImmutable

La classe “**Datetime**”

La classe DateTime est l'outil principal pour manipuler les dates et heures en PHP. Elle permet de créer, modifier, et formater des dates de manière flexible.

Exemple:

```
// Date et heure actuelles
$date = new DateTime();

// Date et heure spécifiques
$date = new DateTime('2024-08-30 14:30:00');
```

La classe “**DatetimeImmutable**”

La classe DateTimeImmutable fonctionne de manière similaire à DateTime, mais elle ne modifie jamais l'objet existant. Toute modification retourne un nouvel objet, ce qui est utile pour éviter des effets de bord.

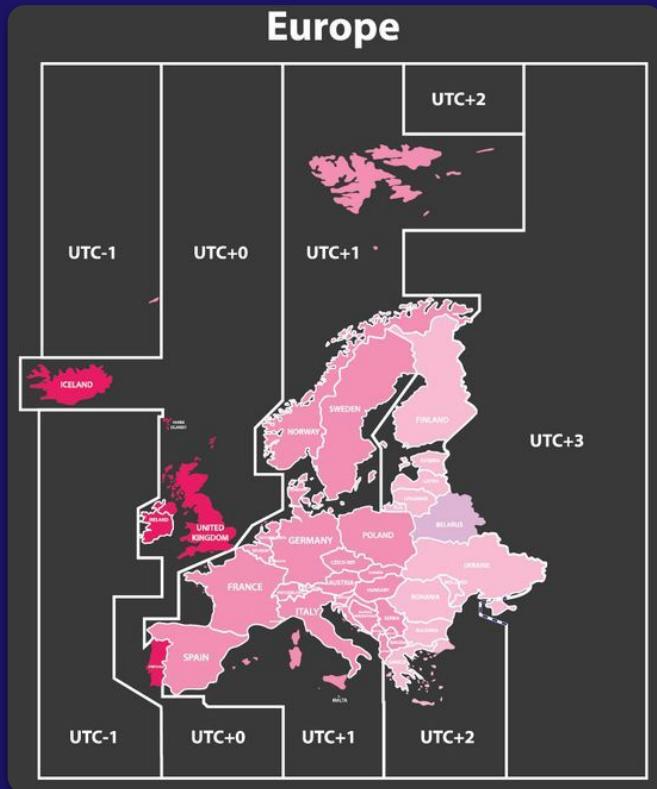
Exemple:

```
// Date et heure spécifiques
$dateImmutable = new DateTimeImmutable('2024-08-30 14:30:00');

// Retourne un nouvel objet
$newDate = $dateImmutable->modify('+1 day');

// Retourne 2024-08-30 14:30:00
$dateImmutable->format('Y-m-d H:i:s');
```

Gestion des Timezones



Les fuseaux horaires

Les fuseaux horaires sont essentiels pour gérer correctement les dates et heures, surtout dans des applications internationales.

```
$timezone = new DateTimeZone(timezone: 'Europe/Paris');
$date = new DateTime(datetime: 'now', $timezone);

// Changer le fuseau horaire
$date->setTimezone(new DateTimeZone(timezone: 'America/New_York'));
```

Voici la constante qui va vous permettre d'identifier toutes les fuseaux horaires disponibles :

```
$timezones = DateTimeZone::listIdentifiers();
```

Gestion des Timezones

Heure d'été / Heure d'hiver

La gestion des heures d'été peut compliquer la manipulation des dates et heures. PHP gère automatiquement les transitions si les fuseaux horaires sont correctement définis.

```
// 1 si en heure d'été, 0 sinon  
$isSummerDate = $date->format(format: 'I');  
  
echo $isSummerDate ? 'Heure d\'été' : 'Heure standard';
```



Bonne pratique avec les timezones

Il est recommandé de stocker les dates en UTC (Coordinated Universal Time) dans les bases de données et de les convertir en fuseau horaire local lors de l'affichage.

Intervalles de Dates

La classe “**DateInterval**”

`DateInterval` est une classe qui représente une période de temps.

Elle est utilisée pour définir des intervalles que vous pouvez ajouter ou soustraire à partir d'un objet `DateTime` ou `DateTimeImmutable`.

```
$date = new DateTime(datetime: '2024-08-30');

// Ajoute 1 mois
$date->add(new DateInterval(duration: 'P1M'));

// Soustrait 10 jours
$date->sub(new DateInterval(duration: 'P10D'));
```

```
// Période de 2 jours
$interval = new DateInterval(duration: 'P2D');

// 1 an, 2 mois, 10 jours
$interval = new DateInterval(duration: 'P1Y2M10D');
```

```
$date1 = new DateTime(datetime: '2024-05-01');
$date2 = new DateTime(datetime: '2023-12-25');
$diff = $date1->diff($date2);

// Affiche : 128 jours
echo $diff->format(format: '%a jours');
```

Périodes de Dates

La classe “**PeriodDate**”

La classe DatePeriod permet de créer un tableau de dates récurrentes entre deux dates données à l'aide de la classe DateInterval.

```
$start = new DateTime(datetime: '2024-01-01');
$end = new DateTime(datetime: '2024-12-31');
$interval = new DateInterval(duration: 'P1M');

$period = new DatePeriod($start, $interval, $end);

foreach ($period as $date) {
    echo $date->format(format: 'Y-m') . "\n";
}
```

```
$start = new DateTime(datetime: '2024-01-01');
$interval = new DateInterval(duration: 'P1M');

$period = new DatePeriod($start, $interval, end: 5);
foreach ($period as $date) {
    echo $date->format(format: 'Y-m');
}
```

Gestion des datetimes

Quelques fonctions utiles

- Créer un Datetime à partir de format non standard.

```
$string = '30/08/2024 14:30';
$date = DateTime::createFromFormat('d/m/Y H:i', $string);
echo $date->format('Y-m-d H:i:s');
```

- Formater un Datetime

```
$date = new DateTime();
echo $date->format('Y-m-d H:i:s'); // Format standard
echo $date->format('d/m/Y'); // Format français
```

- Convertir une date en timestamp UNIX

```
$timestamp = strtotime('2024-12-31');
```

Gestion des datetimes

Quelques fonctions utiles

- Créer un Datetime à partir de format non standard.

```
$string = '30/08/2024 14:30';
$date = DateTime::createFromFormat('d/m/Y H:i', $string);
echo $date->format('Y-m-d H:i:s');
```

- Formater un Datetime

```
$date = new DateTime();
echo $date->format('Y-m-d H:i:s'); // Format standard
echo $date->format('d/m/Y'); // Format français
```

- Convertir une date en timestamp UNIX

```
$timestamp = strtotime('2024-12-31');
```

Le standard ISO 8601

Qu'est que c'est ?

Le standard ISO 8601 est une norme internationale qui spécifie une manière unifiée de représenter les dates et heures.

Ce format est largement utilisé dans les systèmes informatiques, les échanges de données et les bases de données pour garantir que les dates et heures sont représentées de manière cohérente et compréhensible à travers différents systèmes et cultures.

Date complète :

- Format : YYYY-MM-DD
- Exemple : 2024-09-08

Heure complète :

- Format : hh:mm:ss
- Exemple : 14:30:45

Date et heure combinées (UTC) :

- Format : YYYY-MM-DDThh:mm:ssZ
- Exemple : 2024-09-08T14:30:45Z

Date et heure avec décalage horaire :

- 2024-09-08T14:30:45+02:00

Cas pratique



Vous disposez d'une liste d'événements représentant les étapes suivantes d'un projet de développement web.

Les dates de chaque événements sont dans le fuseau horaire correspondant à l'événement.

- Afficher la liste des événements dans un tableau
- Trier les événements par date de début en UTC
- Afficher la durée de chaque événements dans le tableau
 - Exemples: 2 Heures et 20 minutes
- Afficher la date de début et la date de fin en UTC
- Ajouter un menu déroulant pour sélectionner un fuseau horaire dans un menu déroulant.
- Mettez à jour l'affichage des événements en fonction du fuseau horaire sélectionné.

La connexion à la BDD

Avec PDO

```
$serveur = "localhost";
$utilisateur = "nom_utilisateur";
$mot_de_passe = "mot_de_passe";
$base_de_donnees = "nom_de_la_bdd";

try {
    $connexion = new PDO("mysql:host=$serveur;dbname=$base_de_donnees", $utilisateur, $mot_de_passe);
    $connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connexion réussie avec PDO.";
} catch (PDOException $e) {
    die("Échec de la connexion : " . $e->getMessage());
}
```

Avec MySQLi

```
$serveur = "localhost";
$utilisateur = "nom_utilisateur";
$mot_de_passe = "mot_de_passe";
$base_de_donnees = "nom_de_la_bdd";

$connexion = new mysqli($serveur, $utilisateur, $mot_de_passe, $base_de_donnees);

if ($connexion->connect_error) {
    die("Échec de la connexion : " . $connexion->connect_error);
}

echo "Connexion réussie avec MySQLi.";
```

La PHPDoc

```
/**  
 * Divise deux nombres et retourne le résultat.  
 *  
 * Cette fonction prend deux nombres en entrée, effectue une division  
 * et retourne le résultat. Si le diviseur est zéro, une exception est levée.  
 *  
 * @param float $numérateur Le numérateur (le nombre à diviser).  
 * @param float $dénominateur Le dénominateur (le diviseur).  
 * @return float Le résultat de la division.  
 * @throws InvalidArgumentException Si le dénominateur est zéro  
 */  
  
no usages  
  
function diviser(float $numérateur, float $dénominateur): float {  
    if ($dénominateur == 0) {  
        throw new InvalidArgumentException(message: 'Le dénominateur ne peut pas être zéro.');//  
    }  
    return $numérateur / $dénominateur;  
}
```

La PHPDoc est un système de documentation standardisé utilisé pour documenter le code PHP.

Il permet de fournir des informations sur les fonctions, les méthodes, les classes, les variables, etc., sous forme de commentaires structurés. PHPDoc est particulièrement utile pour

- Expliquer le but d'une fonction ou d'une méthode.
- Décrire les paramètres et les valeurs de retour.
- Indiquer les exceptions ou erreurs pouvant être levées.
- Rendre le code plus maintenable

Les constantes magiques

Les constantes magiques en PHP sont des constantes prédéfinies qui changent en fonction de leur contexte d'utilisation. Elles permettent d'obtenir des informations sur des éléments spécifiques du code, comme le nom du fichier, le numéro de la ligne, ou encore la classe où elles sont appelées. Elles sont appelées "magiques" car leurs valeurs sont définies automatiquement par PHP à l'exécution et peuvent changer selon l'endroit où elles sont utilisées.

<code>__LINE__</code>	La ligne courante dans le fichier.
<code>__FILE__</code>	Le chemin complet et le nom du fichier courant
<code>__DIR__</code>	Le nom du dossier courant.
<code>__FUNCTION__</code>	Le nom de la fonction
<code>__CLASS__</code>	Le nom de la classe courante.
<code>__TRAIT__</code>	Le nom du trait.
<code>__METHOD__</code>	Le nom de la méthode courante.
<code>__NAMESPACE__</code>	Le nom de l'espace de noms courant.

Les constantes magiques

Les constantes magiques en PHP sont des constantes prédéfinies qui changent en fonction de leur contexte d'utilisation. Elles permettent d'obtenir des informations sur des éléments spécifiques du code, comme le nom du fichier, le numéro de la ligne, ou encore la classe où elles sont appelées. Elles sont appelées "magiques" car leurs valeurs sont définies automatiquement par PHP à l'exécution et peuvent changer selon l'endroit où elles sont utilisées.

```
no usages
function division($a, $b) {
    if ($b == 0) {
        echo "Erreur dans " . __FUNCTION__ . " à la ligne "
            . __LINE__ . " dans le fichier " . __FILE__ .
            ": Division par zéro.\n";
        return null;
    }
    return $a / $b;
}
```

Les namespaces

Qu'est que c'est ?

Les namespaces en PHP sont une fonctionnalité qui permet d'organiser et de regrouper des classes, des fonctions et des constantes afin d'éviter les conflits de noms dans de grands projets ou lors de l'intégration de bibliothèques externes.

Ils permettent de créer des espaces de noms distincts, qui facilitent la gestion du code dans des projets complexes en évitant que deux classes ou fonctions ayant le même nom ne provoquent des erreurs.

Pour déclarer un namespace dans un fichier PHP, utilisez le mot-clé `namespace` suivi du nom de l'espace de noms. La déclaration doit être faite en haut du fichier, avant toute autre instruction

```
namespace App\Utilisateurs;
```

```
require 'User.php';
```

```
$user = new \MonApp\Utilisateurs\User();
```

```
require 'User.php';
```

```
use MonApp\Utilisateurs\User;
```

```
$user = new User();
```

Fonction Levenshtein



Cette fonction est utilisée pour calculer la distance de Levenshtein entre deux chaînes de caractères.

Cette distance représente le nombre minimal d'opérations nécessaires pour transformer une chaîne en une autre. Les opérations autorisées sont :

- Insertion
- Suppression
- Substitution

Cette fonction est particulièrement utile pour implémenter une recherche floue, où l'utilisateur peut fournir des entrées avec des fautes de frappe ou des variations mineures.

A noter qu'il existe également la fonction 'similar_text()', néanmoins celle-ci est beaucoup plus lourde.

Cas pratique



Vous disposez d'une liste d'utilisateurs. L'objectif de cet exercice est de mettre en pratique la recherche approximative.

- Implémenter une recherche approximative à l'aide de la fonction levenshtein
- Afficher les résultats de la recherche dans un tableau
 - Nom & Prénom
 - Distance avec le nom
 - Distance avec le prénom

Les fonctions récursives

Définition

Une fonction récursive est une fonction qui s'appelle elle-même, directement ou indirectement, dans son propre corps. Cela permet de résoudre des problèmes qui peuvent être divisés en sous-problèmes plus petits du même type.

Exemple:

```
2 usages
function sommeTableau($tableau, $index = 0) {
    if ($index == count($tableau)) {
        return 0;
    } else {
        return $tableau[$index] + sommeTableau($tableau, $index + 1);
    }
}

$tableau = [1, 2, 3, 4, 5];
$somme = sommeTableau($tableau);
echo $somme;
```

Les expressions régulières (regex)

Qu'est ce que c'est ?

Les expressions régulières (ou regex) sont des motifs utilisés pour correspondre à des séquences de caractères dans des chaînes de texte.

En PHP, les expressions régulières sont utilisées pour rechercher, valider, ou remplacer des portions spécifiques de texte, de manière très flexible et puissante.

Pourquoi les utiliser ?

Les regex sont couramment utilisées pour valider des formats de données, comme des adresses email, des numéros de téléphone, des URL, ou encore des codes postaux.

```
$email = "c.haller@ec2e.com";

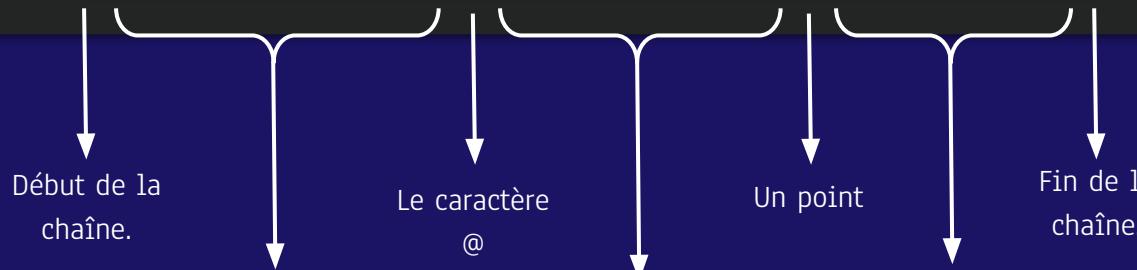
if (preg_match( pattern: '/^[\w\.-%+-]+@[a-zA-Z0-9.-]+\.\w{2,}$/' , $email)) {
    echo 'Email valide.';
} else {
    echo 'Email invalide.';
}
```

Décomposition de la regex

Utilisation de la fonction **preg_match()**

La fonction **preg_match()** permet de vérifier si une chaîne contient un motif spécifique défini par une expression régulière. Cette fonction renvoie 1 si il y a une correspondance sinon 0, elle peut renvoyer false si il y a une erreur de syntaxe.

```
preg_match( pattern: '/^ [a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\. [a-zA-Z]{2,}$/, $email);
```



Un ou plusieurs caractères (lettres, chiffres, point, tiret, etc.)

Un ou plusieurs caractères

Deux lettres minimum ou plus pour le suffixe de domaine

Les fonctions utilisant les regex

La fonction **preg_match_all()**

La fonction **preg_match_all()** permet de trouver toutes les occurrences d'un motif dans une chaîne et de les stocker dans un tableau.

La fonction **preg_replace()**

La fonction **preg_replace()** permet de remplacer toutes les occurrences d'un motif par une autre chaîne.

La fonction **preg_split()**

La fonction **preg_split()** permet de diviser une chaîne en plusieurs sous-chaînes en utilisant une expression régulière comme délimiteur. Cette fonction retourne un tableau.

La fonction **preg_grep()**

La fonction **preg_grep()** permet de filtrer les éléments d'un tableau qui correspondent ou non à un motif.

La manipulation de fichiers

En PHP, la manipulation des fichiers permet de lire, écrire, créer, et supprimer des fichiers sur le système de fichiers du serveur. PHP fournit plusieurs fonctions pour travailler avec des fichiers de manière efficace.

Récupérer le contenu d'un fichier :

```
// Ouvre le fichier en lecture
$fichier = fopen( filename: 'mon_fichier.txt', mode: 'r' );
// Lit tout le fichier
$contenu = fread($fichier, filesize( filename: 'mon_fichier.txt' ));
// Affiche le contenu du fichier
echo $contenu;
// Ferme le fichier
fclose($fichier);
```

Est égale à:

```
$contenu = file_get_contents( filename: 'mon_fichier.txt' );
echo $contenu;
```

Récupérer le contenu d'un fichier ou d'un URL

La fonction **file_get_contents()**

Cette fonction prend une URL ou un chemin de fichier en paramètre et renvoie le contenu sous forme de chaîne de caractères.

Si l'URL est correcte et que tout fonctionne bien, vous obtiendrez tout le HTML de la page.

```
$url = 'https://www.example.com';
$htmlContent = file_get_contents($url);

if ($htmlContent === false) {
    echo "Erreur lors de la récupération du contenu.";
} else {
    echo $htmlContent;
}
```

Le fichier robot.txt

Le fichier **robots.txt** est un fichier texte placé à la racine d'un site web qui donne des instructions aux robots d'exploration (ou "**crawlers**") des moteurs de recherche, comme Google, Bing, ou d'autres services.

Ce fichier est utilisé pour contrôler ce que les robots peuvent explorer et indexer sur votre site.

```
User-agent: Googlebot  
Disallow: /no-google/
```

```
User-agent: Bingbot  
Disallow: /no-bing/
```

```
User-agent: *  
Disallow: /private/
```

```
User-agent: *  
Disallow: /admin/  
Disallow: /private/  
Allow: /public/
```

```
Sitemap: https://www.exemple.com/sitemap.xml
```

- User-agent : Spécifie à quel robot la règle s'applique.
- Disallow : Indique les URL que les robots ne doivent pas explorer.
- Allow : Indique les URL que les robots peuvent explorer (souvent utilisé pour autoriser des sous-pages spécifiques).
- Sitemap : Spécifie l'emplacement du fichier sitemap du site (utile pour les moteurs de recherche pour découvrir toutes les URL à explorer).

Le fichier sitemap XML

Un fichier **sitemap XML** est un fichier qui répertorie toutes les URL importantes d'un site web.

Ce fichier aide les moteurs de recherche comme Google à explorer et à indexer efficacement un site en fournissant une liste structurée des pages, avec des informations supplémentaires telles que la date de dernière modification, la priorité de chaque page et la fréquence de mise à jour.

NB: Le fichier sitemap XML est un élément clé du référencement (SEO), car il permet aux moteurs de recherche de découvrir et d'indexer les pages importantes d'un site de manière efficace. Il fournit des informations précieuses telles que la fréquence des mises à jour des pages, leur importance relative, et la date de leur dernière modification.

```
xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">

    <!-- Page d'accueil -->
    <url>
        <loc>https://www.exemple.com/</loc>
        <lastmod>2023-01-01</lastmod>
        <changefreq>daily</changefreq>
        <priority>1.0</priority>
    </url>

    <!-- Page de contact -->
    <url>
        <loc>https://www.exemple.com/contact</loc>
        <lastmod>2023-01-01</lastmod>
        <changefreq>monthly</changefreq>
        <priority>0.8</priority>
    </url>

    <!-- Article de blog -->
    <url>
        <loc>https://www.exemple.com/blog/article-1</loc>
        <lastmod>2023-02-15</lastmod>
    </url>
```

Cas pratique : Création d'un “scrapper”



Créez une page web en PHP qui permet de récupérer toutes les adresses email présentes sur une page donnée.

La page doit contenir :

- Un formulaire HTML avec un champ texte pour entrer une URL.
- Un tableau pour afficher la liste des adresses Email.

Contraintes :

- Vérifiez si l'URL saisie est dans un format valide en utilisant une regex.
- Utilisation de Regex pour récupérer les adresses Email
- Gérer les erreurs si l'URL n'existe pas ou est incorrect.

Création d'un “scrapper” avancé



Créez une page web en PHP qui permet d'extraire toutes les adresses email d'une URL donnée, mais aussi de parcourir récursivement les autres pages liées à cette URL (c'est-à-dire toutes les pages web référencées par des liens dans la page HTML initiale), et d'extraire les adresses email trouvées dans chacune de ces pages.

La page doit contenir :

- Un formulaire HTML avec un champ texte pour entrer une URL.
- Un tableau pour afficher la liste des adresses Email.
- Un tableau pour afficher toutes les URL visitées.

Contraintes supplémentaire:

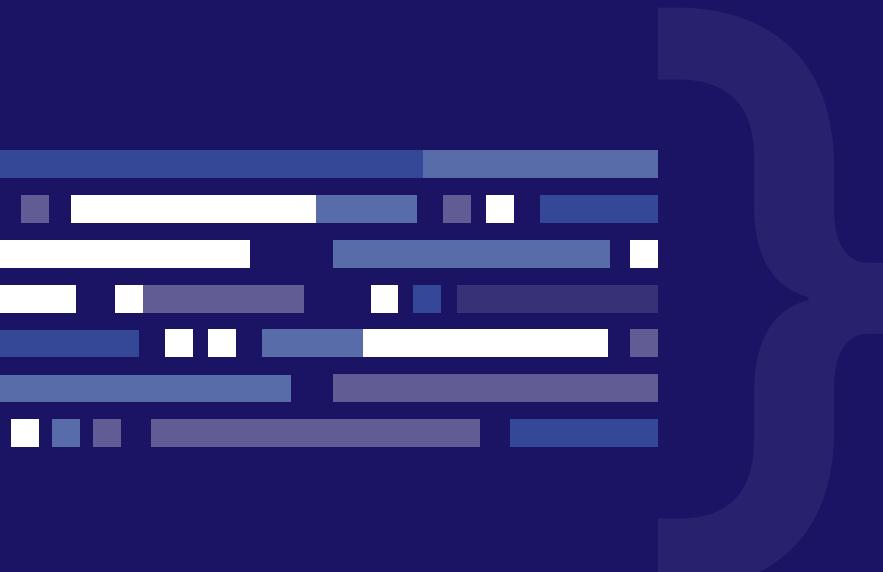
- Récupérez récursivement toutes les pages liées depuis l'URL initiale (liens trouvés dans les balises ``).
- Limiter la récursion à 10 niveaux

</>

Cours de POO PHP

May the force be with you !

LA POO PHP



- Construire une classe
- Instancier un objet
- Propriétés & Méthodes
- Constructeur
- Public & Privé
- Accesseurs et Mutateurs
- Propriétés & Méthodes statiques
- Héritage
- Traits
- Les classes abstraites
- Les méthodes magiques
- Documenter une classe

LA STRUCTURE MVC

Définition

Les avantages d'utiliser ce Design Pattern

L'arborescence d'un projet MVC

Le contrôleur frontal / Routeur

Les requête HTTP

La redirection vers un contrôleur

Analyse du routage d'une URL

La connexion à une BDD

La gestion des Modèles

Comment éviter les injections SQL

Création d'un formulaire

Le softDelete

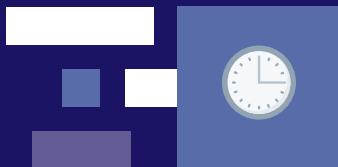
Authentification & Inscription

Réalisation d'un POC sur l'ensemble des points vue en cours

Qu'est ce que c'est que la POO ?

La Programmation Orientée Objet (POO) est un paradigme de programmation qui organise le code en utilisant des objets plutôt que des fonctions et des procédures.

Elle vise à modéliser des entités du monde réel sous forme de classes et d'objets, facilitant ainsi la maintenance, la réutilisabilité et la scalabilité du code.

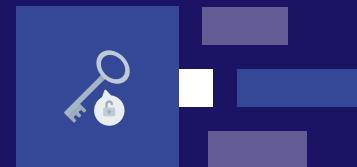


Maintenabilité

- Le code est organisé
- Structure définie

Gain de temps

Code réutilisable
Modélisation Réaliste



Construire une classe

```
class Voiture {  
  
    public $marque;  
    public $couleur;  
  
    public function start() {  
        echo "La voiture démarre.";  
    }  
}
```

Une « classe » est un modèle de données définissant la structure commune à tous les objets qui seront créés à partir d'elle.

Plus concrètement, nous pouvons percevoir une classe comme un moule grâce auquel nous allons créer autant d'objets de même type et de même structure qu'on le désire.

Instancier un objet

```
require 'Voiture.php';

// Création d'un objet "Voiture"
// On dit que l'on crée une instance
// de la classe Voiture
$car = new Voiture();
```

Une instance est une représentation particulière d'une classe.

Lorsque l'on crée un objet, on réalise ce que l'on appelle une « instance de la classe ».

C'est à dire que du moule, on en extrait un nouvel objet qui dispose de ses attributs et de ses méthodes. L'objet ainsi créé aura pour type le nom de la classe.

Propriétés & Méthodes

```
require 'Voiture.php';

$car = new Voiture();

$car->marque = "Tesla";
$car->couleur = "Blanche";
```

Ici on définit une valeur pour les propriétés de notre objet.

Il est possible de définir une valeur à une propriété inexistante dans la classe.

Néanmoins cela est déconseillé afin de conserver un code maintenable.

Propriétés & Méthodes

```
require 'Voiture.php';

$car = new Voiture();

$car->marque = "Tesla";
$car->couleur = "Blanche";
$car->place = 5;
```

Ici on définit une valeur pour les propriétés de notre objet.

Il est possible de définir une valeur à une propriété inexistante dans la classe.



Néanmoins cela est déconseillé afin de conserver un code maintenable. (Dynamic properties :Non maintenable en PHP 9)

Propriétés & Méthodes

```
class Voiture {  
  
    public $marque;  
    public $couleur;  
  
    public function start() {  
        echo "La $this->marque $this->couleur démarre.";  
    }  
}
```

\$this est un mot-clé spécial en PHP utilisé à l'intérieur d'une classe pour faire référence à l'instance courante de cette classe, c'est-à-dire l'objet lui-même.

Lorsque vous travaillez au sein d'une méthode d'une classe, **\$this** vous permet d'accéder aux propriétés et méthodes de l'objet spécifique qui a appelé cette méthode.

Constructeur

```
class Voiture {  
  
    public $marque;  
    public $couleur;  
  
    public function __construct($marque, $couleur)  
    {  
        $this->marque = $marque;  
        $this->couleur = $couleur;  
    }  
  
    public function start()  
    {  
        echo "La $this->marque $this->couleur démarre.";  
    }  
}
```

Le constructeur est une méthode particulière. C'est celle qui est appelée implicitement à la création de l'objet (instanciation).

Le développeur est libre de définir des paramètres obligatoires à passer au constructeur ainsi qu'un groupe d'instructions à exécuter à l'instanciation de la classe.

```
require 'Voiture.php';  
  
$car = new Voiture( marque: "Tesla", couleur: "Blanche");
```

Public & Private

```
class Voiture {  
  
    public $marque;  
    public $couleur;  
    private $kilometres;  
  
    public function __construct($marque, $couleur, $kilometres)  
    {  
        $this->marque = $marque;  
        $this->couleur = $couleur;  
        $this->kilometres = $kilometres;  
    }  
  
    public function start()  
    {  
        echo "La $this->marque $this->couleur démarre.";  
    }  
}
```

Le mot-clé ***public*** permet de rendre l'attribut accessible depuis l'extérieur de la classe.

Le mot-clé ***private*** permet de rendre l'attribut accessible depuis l'intérieur de la classe.

Ici l'attribut "kilometres" n'est plus accessible en dehors de la classe.

```
require 'Voiture.php';  
  
$car = new Voiture( marque: "Tesla", couleur: "Blanche", kilometres: 200000);  
  
$car->kilometres;
```

Accesseurs et Mutateurs

```
class CompteBancaire {  
    private $capital = 0;  
  
    public function set($montant) {  
        $this->capital += $montant;  
    }  
  
    public function take($montant) {  
        if ($montant <= $this->capital) {  
            $this->capital -= $montant;  
        } else {  
            echo "Fonds insuffisants."  
        }  
    }  
  
    public function displayCapital() {  
        echo "Le solde est de : " . $this->capital . " €."  
    }  
}
```

En règle générale, on n'accède pas directement aux attributs d'un objet.

Pour lire et modifier leurs valeurs, on passe par des méthodes qui permettent de sécuriser leur utilisation.

Une des conventions souvent utilisée est de reprendre le nom de l'attribut pour créer les méthodes, en ajoutant `get` pour les accesseurs et `set` pour les mutateurs.

La norme d'encapsulation :

L'encapsulation consiste à regrouper les données (propriétés) et les méthodes qui manipulent ces données au sein d'une même classe, tout en contrôlant l'accès aux données via des modificateurs d'accès.

Propriétés & Méthodes statiques

```
class CompteBancaire {  
    private $capital = 0;  
    private static $overdraft = -400;  
  
    public function getOverdraft(){  
        return $this->overdraft;  
    }  
  
    public function set($montant) {...}  
  
    public function take($montant) {...}  
  
    public function displayCapital() {...}  
}
```

Les propriétés et les méthodes statiques peuvent être utilisées sans avoir besoin d'instancier la classe.

On peut y accéder directement en utilisant le nom de la classe.

```
require 'CompteBancaire.php';  
  
$overdraft = CompteBancaire::getOverdraft();
```

Héritage

```
class Banker extends User
{
    public $bank = 'BNP';

    public static $min_money = -800;

    public function __construct($name = '', $firstname = '', $money = 0, $bank = 'BNP')
    {
        $this->bank = $bank;
        parent::__construct($name, $firstname, $money);
    }
}
```

```
$johnson = new Banker(name: 'Johnson', firstname: 'Philippe', money: 2000, bank: 'CIC');
var_dump($johnson);
```



```
✓ Banker (object) [Object ID #2][4 properties]
  bank: (string) "CIC"
  name: (string) "Johnson"
  firstname: (string) "Philippe"
  money:User:private: (integer) 2000
```

L'héritage en PHP permet de créer une nouvelle classe qui héritera des propriétés et des méthodes d'une classe parent et qui pourra, si on le souhaite, redéfinir certaines propriétés et méthodes.

Afin d'hériter d'une classe, il faut utiliser le mot clés "**extends**".

Le Polymorphisme

Qu'est ce que c'est ?

Lorsque des classes héritent d'une classe parente commune et redéfinissent certaines méthodes, on peut utiliser les objets de ces classes de manière polymorphe.

En POO, cela signifie que le même morceau de code peut fonctionner avec différents types d'objets.

Le polymorphisme améliore la flexibilité et l'extensibilité du code, facilitant ainsi la maintenance et l'ajout de nouvelles fonctionnalités sans modifier le code existant.

```
class Animal {  
    public function communiquer() {  
        echo "L'animal communique.\n";  
    }  
}  
  
class Chien extends Animal {  
    public function communiquer() {  
        echo "Le chien aboie.\n";  
    }  
}  
  
class Chat extends Animal {  
    public function communiquer() {  
        echo "Le chat miaule.\n";  
    }  
}
```

Les Interfaces

```
interface Animal {  
    public function communiquer();  
}  
  
class Chien implements Animal {  
    public function communiquer() {  
        echo "Le chien aboie.\n";  
    }  
}  
  
class Chat implements Animal {  
    public function communiquer() {  
        echo "Le chat miaule.\n";  
    }  
}
```

Qu'est ce que c'est ?

Une interface est un contrat qui définit un ensemble de méthodes publiques qu'une classe doit implémenter.

Les interfaces permettent de garantir que différentes classes partagent les mêmes méthodes, même si elles n'ont pas de lien d'héritage direct.

Cela facilite le polymorphisme, où des objets de classes différentes peuvent être traités de la même manière s'ils implementent la même interface.

Les classes abstraites

```
abstract class AbstractClass
{
    protected $purchaseOrder;
    protected $amount;
    protected $price;

    abstract protected function getOrderInfo();
}

class Car extends AbstractClass
{
    protected function getOrderInfo() {
        return "Some Info";
}
}
```

PHP a des classes et des méthodes abstraites.

Les classes définies comme abstraites ne peuvent pas être instanciées et toute classe contenant au moins une méthode abstraite doit également être abstraite.

Les classes abstraites sont souvent utilisées lors de cas d'héritage.

Les Traits

Un trait est semblable à une classe, mais il ne sert qu'à grouper des fonctionnalités d'une manière intéressante. Il n'est pas possible d'instancier un Trait en lui-même.

```
trait Inventaire
{
    public $nombre;

    public function plusUn()
    {
        $this->nombre++;
        echo $this->nombre . '<br>';
        return $this;
    }
}
```

Les traits permettent d'éviter une redondance de code dans les classes.

On l'appelle comme ci-dessous.

```
use Inventaire;
```

Le but de cet exercice est de créer une classe PHP (avec les propriétés privées '\$text' & '\$length') au nom de Text qui contient les méthodes du même nom que 'String' en Javascript.

Exercice

length(): Cet attribut retourne le nombre de caractères contenus dans la chaîne.

charAt(): La méthode charAt(x) permet de retourner le caractères qui se trouve à la position x passé en paramètre. Le paramètre est un entier qui commence de 0.

indexOf(): La méthode indexOf(car)permet de retourner la position du caractères car passé en paramètre. Si le caractère existe dans la chaîne, alors sa position (comprise entre 0 et la longueur de la chaîne - 1) est retournée, sinon (le caractère ne figure pas dans la chaîne) alors la valeur -1 est retournée. La méthode indexOf() peut accueillir un deuxième paramètre qui est un entier qui indique à partir de quel position de la chaîne on commencera la recherche du caractère passé en premier paramètre.

substring(): La méthode substring(début,fin) permet d'extraire une partie de la chaîne de caractères commençant de la position début et finissant à la position fin-1.

split(): La méthode split(str) permet de retourner un tableau à partir des fractions de la chaîne de caractères obtenues en divisant celle-ci au niveau de 'str'.

toLowerCase(): La méthode toLowerCase() permet de retourner la chaîne de caractère entièrement en minuscules.

toUpperCase(): La méthode toUpperCase() permet de retourner la chaîne de caractère entièrement en majuscules.

Solution

```
class Text
{
    private $text;
    private $length;

    public function __construct($chaine){
        $this->text=$chaine;
        $this->length=strlen($this->text);
    }

    public function setText($text){
        return $this->text = $text;
    }

    public function getText(){
        return $this->text;
    }

    public function getLength(){
        return $this->length;
    }

    public function charAt($pos){
        return substr($this->text,$pos, length: 1);
    }

    public function indexOf($car,$pos=0){
        $existe="non";
        for($i=0;$i<$this->length;$i++){
            if(substr($this->text,$i, length: 1)==$car && $i>=$pos){
                $existe="oui";
                return $i;
            }
        }
        if($existe=="non")
            return -1;
    }
}
```

```
public function substring($deb,$fin){
    return substr($this->text,$deb, length: $fin-$deb);
}

public function split($occ){
    $tab=explode( separator: "$occ", $this->text);
    return $tab;
}

public function toUpperCase(){
    return strtoupper($this->text);
}

public function toLowerCase(){
    return strtolower($this->text);
}
```

Le clonage d'objet

Pourquoi cloner ?

lorsque vous affectez un objet à une nouvelle variable, vous ne créez pas une copie indépendante de cet objet, mais plutôt une référence à l'objet original. Cela signifie que toute modification apportée à l'objet via l'une des références sera reflétée dans toutes les autres références.

```
$laura = new User( name: 'SMET', firstname: 'Laura');  
$david = $laura;  
$david->firstname = 'David';  
  
var_dump($laura);
```



Le clonage d'objet

Comment cloner ?

Vous pouvez utiliser le mot-clé `clone` pour créer une nouvelle instance de l'objet avec les mêmes valeurs pour toutes les propriétés de l'objet original, mais les deux objets sont totalement indépendants l'un de l'autre.

```
$laura = new User( name: 'SMET', firstname: 'Laura');
$david = clone $laura;
$david->firstname = 'David';

var_dump($laura);
```



Documenter une classe

Le fait de documenter vos classes va contribuer à rendre votre code plus maintenable et surtout plus facile d'utilisation pour autrui.

Ainsi l'IDE dont vous vous servez pour utiliser cette documentation afin d'orienter votre façon de développer.

```
/**  
 * Class User  
 * Permet de créer un nouvel utilisateur  
 */  
class User  
{  
    /**  
     * @var String Ceci est le prénom  
     */  
    private $firstname;  
  
    /**  
     * @param $name User Ceci est le nom  
     * @param $firstname User Ceci est le prénom  
     * @param $email User Ceci est une adresse Email  
     */  
    public function __construct($name = null, $firstname = null, $email = null){  
        $this->name = $name;  
        $this->firstname = $firstname;  
        $this->email = $email;  
    }  
    /**  
     * @return string Retourne le prénom et le nom  
     */  
    public function __toString(){  
        return $this->firstname." ".$this->name;  
    }  
}
```

Les méthodes et surcharges magiques

Les méthodes magiques sont des méthodes spéciales qui écrasent l'action par défaut de PHP quand certaines actions sont réalisées sur un objet.

Voici quelques exemples de ces méthodes:

<code>__construct</code>	<code>__toString</code>
<code>__sleep</code>	<code>__debugInfo</code>
<code>__destruct</code>	<code>__clone</code>
<code>__set</code>	<code>__get</code>
<code>__unset</code>	<code>__isset</code>



Toutes les méthodes commençant par `__` sont réservées par PHP. Ainsi, il n'est pas recommandé d'utiliser un tel nom de méthode sauf lors de l'écrasage du comportement de PHP

</>

Structure MVC

Définition du modèle MVC

Le modèle MVC est un modèle de conception, ou un "Design pattern". Il permet de définir l'architecture d'une application qui divise une application en trois composants principaux :

Le Modèle (Model)

- Le modèle représente les données de l'application et la logique métier.
- Il est responsable de gérer l'accès aux données, de les manipuler et de fournir des interfaces pour les récupérer ou les mettre à jour.
- Le modèle n'a pas connaissance de la manière dont les données sont présentées à l'utilisateur.

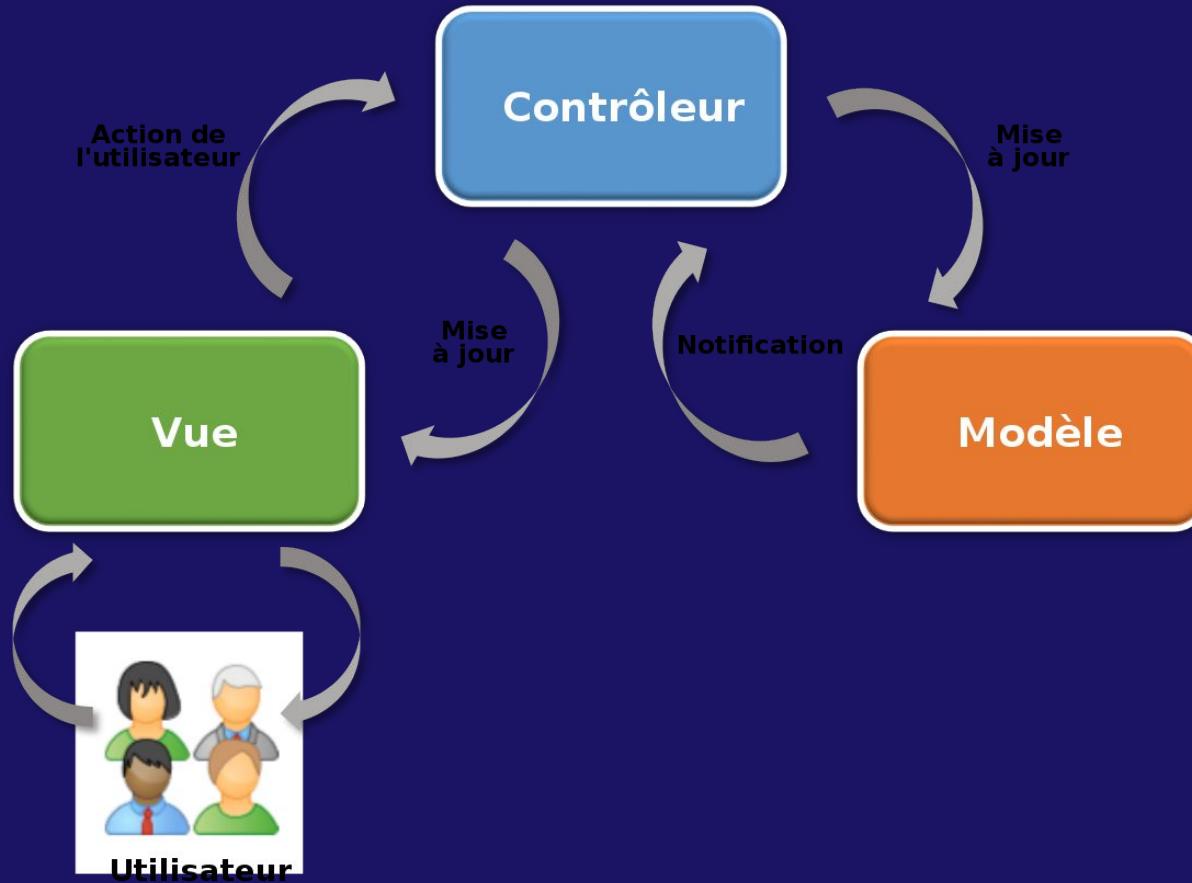
Le Vue (View)

- Elle se charge de l'affichage des données et de l'interaction avec l'utilisateur.
- Elle se contente d'afficher les données et de transmettre les actions de l'utilisateur au contrôleur.

Le Contrôleur (Controller)

- Il gère les actions de l'utilisateur, interagit avec le modèle pour récupérer ou mettre à jour les données, puis met à jour la vue en conséquence.
- Il contient la logique d'orchestration de l'application.

Les interactions



Pourquoi utiliser cette structure ?

Le modèle MVC (Modèle-Vue-Contrôleur) offre plusieurs avantages dans le développement d'applications, en particulier dans le contexte des applications web. Voici quelques-uns des principaux avantages :

Gain de temps

- Composants réutilisables
- Structure définie

Sécurité

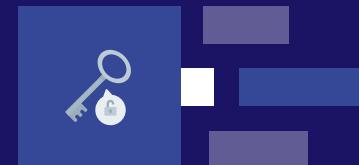
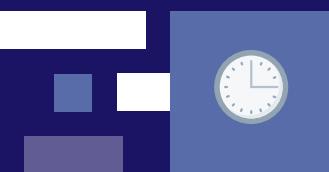
Inclut plusieurs mesures de sécurité pour protéger les applications web des attaques courantes.

Maintenabilité

- Facilité de Modification
- Évite la duplication de code.

Framework friendly

Cela facilite l'apprentissage de nombreux Framework PHP

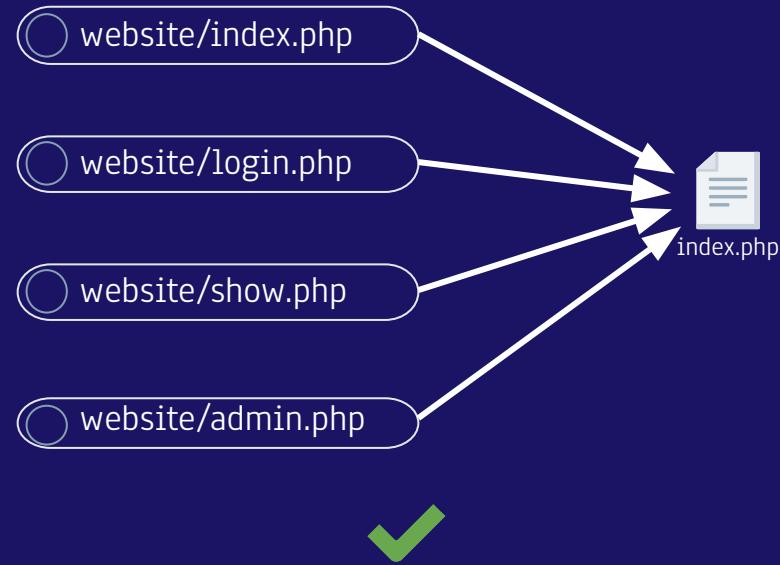
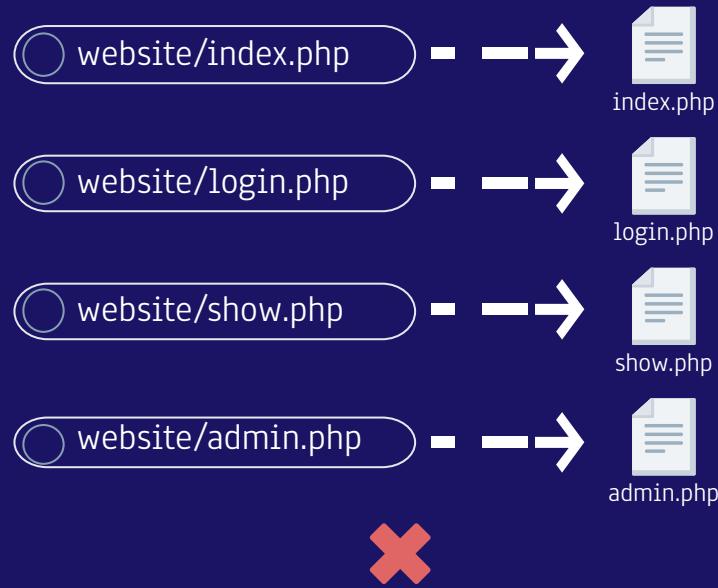


Exemple de framework utilisant cette architecture : Symfony, Laravel, CakePHP ...

Contrôleur Frontal / Routeur

Le contrôleur frontal est le point central de l'application qui dirige la requête initiale vers le bon contrôleur et l'action correspondante. Cela permet d'organiser le code de manière à ce qu'il soit facile à gérer et à comprendre.

Dans le développement web, on utilise le fichier "**index.php**" comme Routeur.



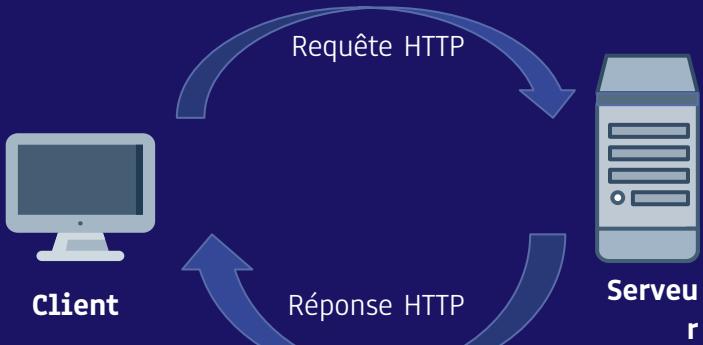
Les requêtes HTTP

Rappel :

Une requête HTTP (HyperText Transfer Protocol) est un message envoyé à un serveur Web pour demander une action ou une information. Elles peuvent être utilisées pour afficher des pages Web, soumettre des formulaires en ligne, envoyer des données à une API, etc.

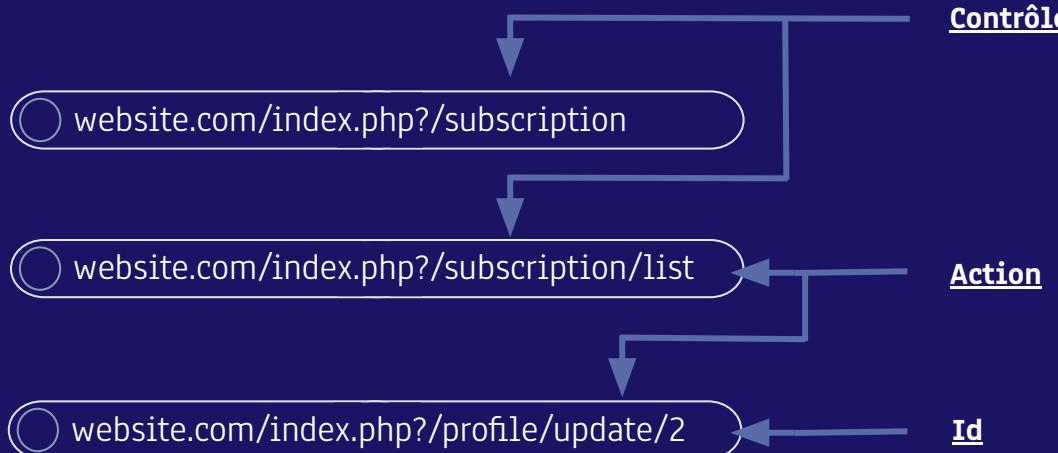
Une requête HTTP se compose généralement de plusieurs parties, telles que :

- Le verbe HTTP (GET, POST, PUT, DELETE, etc.), qui définit l'action demandée
- L'URL cible, qui spécifie la ressource demandée
- Les en-têtes HTTP, qui fournissent des informations supplémentaires sur la requête, telles que le type de contenu accepté ou la préférence de langue
- Le corps de la requête, qui peut contenir des données supplémentaires envoyées avec la requête, telles que les données d'un formulaire Web.



Redirection vers un contrôleur

Afin de rediriger notre route vers un contrôleur, il faut analyser l'URL et déterminer quel contrôleur doit être appelé et quelle action doivent être appelés.



Les étapes de la redirection :

- Envoie de la requête HTTP
- Le routeur reçoit la requête
- Analyse la l'URL (Routage)
- Invoque le contrôleur appelé
- Le contrôleur traite la requête
- Le contrôleur retourne une réponse

NB: Afin de récupérer les éléments de votre URL qui correspondent à ce que vous recherchez, vous pouvez utiliser la superglobale "**`$_SERVER`**".

Dans le cas ci-contre:

`$_SERVER['QUERY_STRING'] = '/profile/update/2'`

Analyse & Routage de l'URL

Récupération de l'emplacement du projet



```
define('ROOT', str_replace( search: 'index.php', replace: '', $_SERVER['SCRIPT_FILENAME']));  
session_start();
```

Récupération du contrôleur et de l'action à appeler



```
if (isset($_GET['action']) && !empty($_GET['action'])) {  
    $params = explode( separator: "/", $_GET['action']);  
  
    if ($params[0] != "") {  
        $controller = $params[0];  
        $action = isset($params[1]) ? $params[1] : 'library';  
        $controllerFile = ROOT . 'controllers/' . $controller . 'Controller.php';  
  
        if (file_exists($controllerFile)) {  
            require_once($controllerFile);  
  
            if (function_exists($action)) {  
                if (isset($params[2]) && isset($params[3])) {  
                    $action($params[2], $params[3]);  
                } elseif (isset($params[2])) {  
                    $action($params[2]);  
                } else {  
                    $action();  
                }  
            } else {  
                header( header: 'HTTP/1.0 404 Not Found');  
                require_once('views/errors/404.html');  
            }  
        } else {  
            header( header: 'HTTP/1.0 404 Not Found');  
            require_once('views/errors/404.html');  
        }  
    } else {  
        require_once('controllers/BookController.php');  
        library();  
    }  
}
```

Si le contrôleur et l'action existe, on importe le contrôleur et on appelle l'action



Si le contrôleur ou l'action n'existe pas, je renvoie vers une page 404 (Page non trouvée)



Si l'URL est vide, je renvoie vers une page d'accueil



Réécriture de l'URL

Il s'agit ici de modifier notre URL affiché dans le navigateur.

Pourquoi modifier notre URL ? :

- Cela permet d'améliorer la lisibilité de l'URL.
- Cela peut contribuer à un meilleur classement dans les résultats de recherche (SEO).
- Cela facilite la navigation des utilisateurs.
- Cela permet de masquer les détails techniques de votre site, ce qui peut contribuer à la sécurité en cachant les informations sensibles.

Pour ce faire, nous allons passer une logique de routage personnalisée à l'aide du fichier
".htaccess" (pour Apache)

```
RewriteEngine on  
RewriteRule ^home$ index.php
```

```
RewriteEngine on  
RewriteRule ^([a-zA-Z0-9\-\_\_\/\]*)$ index.php?action=$1
```

NB: Il peut arriver que le contenu du fichier ".htaccess" ne soit pas pris en compte. Vérifier dans le fichier de Apache "http.conf" que AllowOverride est bien None.

L'arborescence

Voici l'arborescence minimum que votre projet devra suivre pour correspondre à la structure MVC.

Ceci est une structure de base, en fonction de vos besoins, vous pourrez l'adapter en ajoutant ou en modifiant des dossiers et des fichiers.



Nous avons déjà vu la plupart de ces éléments précédemment.



Ce dossier va vous permettre de stocker des ressources statiques telles que des fichiers CSS, JavaScript, des images, des polices de caractères, des icônes ...

La connexion à la BDD



Afin de mettre en place la connexion à notre base de données créés au préalable, nous allons créer le fichier "db_connect.php" dans le dossier "models", car c'est dans ces derniers qu'il sera appelé. Pour l'exemple qui suit nous nous connectons à notre base de données en utilisant PDO.

db_connect.php :

```
no usages
function connection(){
    $serveur = "localhost";
    $utilisateur = "root";
    $mot_de_passe = "";
    $base_de_donnees = "mvc";

    try {
        $connexion = new PDO( dsn: "mysql:host=$serveur;dbname=$base_de_donnees", $utilisateur, $mot_de_passe);
        $connexion->setAttribute( attribute: PDO::ATTR_ERRMODE, value: PDO::ERRMODE_EXCEPTION);

        return $connexion;
    } catch (PDOException $e) {
        die("Échec de la connexion : " . $e->getMessage());
    }
}
```

Création de vos modèles



Afin d'organiser et de structurer le code de manière efficace, il est fortement recommandé d'utiliser des classes afin de gérer vos modèles.

Voici les étapes à suivres pour créer vos classes correspondant à vos modèles:

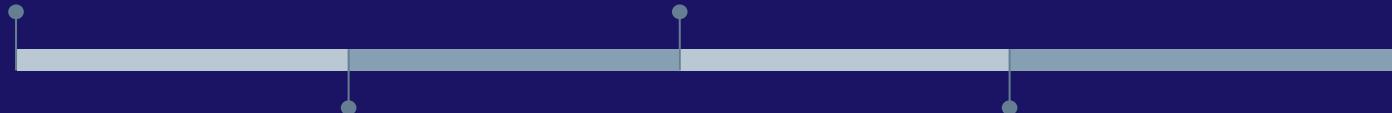
Déclarer votre classe

Créer les méthodes d'accès
(getters et setters)

Ajouter les méthodes liées
aux opérations CRUD

Définir les propriétés de
la classe

Importer la connexion à la
base de données



Création d'une requête SQL “INSERT”



Comme nous venons de le voir dans la création d'un modèle, dans une application PHP basée sur l'architecture MVC, les requêtes SQL doivent principalement être placées dans la partie du Modèle.

Nous allons donc commencer par la fonction create du CRUD et faire une fonction pour ajouter un élément à notre base de données avec un “INSERT”.

Exemple:

```
no usages
public function create($title, $author, $published_year) {
    $db = connection();

    $query = "INSERT INTO books (title, author, published_year, created_at, updated_at)
              VALUES ('$title', '$author', $published_year, NOW(), NOW())";

    $db->exec($query);
}
```

NB: N'oubliez pas d'importer la connexion à votre base de données.

L'injection SQL



L'injection SQL est une technique d'attaque informatique qui vise à manipuler une base de données en injectant du code SQL malveillant à travers des entrées utilisateur non vérifiées ou mal protégées.

Pour éviter les injections SQL, il est crucial d'utiliser des requêtes préparées et des requêtes paramétrées, ainsi que de valider et échapper les entrées utilisateur correctement. Cela empêchera l'injection de code SQL malveillant.

Voici à quoi devrait ressembler notre INSERT maintenant que nous évitons les injections SQL :

```
no usages
public function add($title, $author, $published_year) {
    $db = connection();
    $stmt = $db->prepare( query: 'INSERT INTO books (title, author, published_year, created_at, updated_at)
                                VALUES (:title, :author, :published_year, NOW(), NOW())');

    $stmt->bindValue( param: ':title', $title, type: PDO::PARAM_STR);
    $stmt->bindValue( param: ':author', $author, type: PDO::PARAM_STR);
    $stmt->bindValue( param: ':published_year', $published_year, type: PDO::PARAM_INT);

    $stmt->execute();
}
```



Article sur les
Injection SQL

Création d'un formulaire d'ajout



À l'intérieur du dossier views, créez un fichier PHP pour le formulaire qui permettra d'appeler le contrôleur pour ajouter un livre.

```
<h1>Ajouter un livre</h1>
<form action="/add" method="post">
    <label for="bookTitle">Titre:</label>
    <input type="text" id="bookTitle" name="bookTitle" required><br>

    <label for="bookAuthor">Auteur:</label>
    <input type="text" id="bookAuthor" name="bookAuthor" required><br>

    <label for="bookYear">Année de publication:</label>
    <input type="text" id="bookYear" name="bookYear" required><br>

    <input type="submit" value="Ajouter">
</form>
```

Création d'un contrôleur



Tout d'abord donnez à votre contrôleur un nom qui reflète sa responsabilité. Par exemple, si votre contrôleur gère les livres, nommez-le BookController.php.

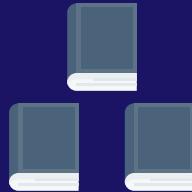
Le rôle du contrôleur est de gérer les requêtes HTTP et appeler les méthodes appropriées du modèle en fonction de l'action demandée. N'oubliez pas d'importer les modèles que votre contrôleur va utiliser.

```
no usages
function add(){
    if (isset($_POST['bookTitle'], $_POST['bookAuthor'], $_POST['bookYear'])) {
        $title = $_POST['bookTitle'];
        $author = $_POST['bookAuthor'];
        $published_year = $_POST['bookYear'];

        if (!empty($title) && !empty($author) && !empty($published_year)) {
            $book = new Book( id: null, $title, $author, $published_year, created_at: null, updated_at: null);
            $book->create($title, $author, $published_year);
            echo "Livre ajouté avec succès!";
        } else {
            echo "Tous les champs doivent être remplis.";
        }
    }

    require_once ('views/book/form.php');
}
```

Récupération de plusieurs éléments



Maintenant que nous savons gérer nos vues, notre contrôleur et notre modèle. Nous allons faire une page pour récupérer l'ensemble des livres.

Nous allons commencer par construire notre requête dans le modèle :

models/Book.php

```
1 usage
public static function getBooks(){
    try {
        $db = connection();
        $stmt = $db->prepare( query: "SELECT * FROM books ORDER BY published_year ASC");
        $stmt->execute();
        $books = $stmt->fetchALL( mode: PDO::FETCH_ASSOC);
        return $books;
    } catch (PDOException $e) {
        die('Erreur de requête : ' . $e->getMessage());
    }
}
```

controller/BookController.php

```
no usages
function library(){
    $books = Book::getBooks();

    require_once ('views/book/library.php');
}
```

Affichage de plusieurs éléments



Maintenant nous allons construire la vue qui va afficher la liste de livres :

`<?=...?>` est une syntaxe courte qui équivaut à `<?php echo ... ?>`. Elle est utilisée pour afficher directement une variable ou une expression dans du HTML.

views/book/library.php

```
<div class="container">
    <h2 class="text-center">Liste des Livres (total: <?= count($books) ?>)</h2>
    <div class="row" style="margin-top: 2rem">
        <?php foreach ($books as $book): ?>
            <div class="col-md-4">
                <div class="panel panel-default">
                    <div class="panel-body text-center">
                        <span class="glyphicon glyphicon-book" style="font-size: 50px;"></span>
                        <h4><?= $book['title'] ?></h4>
                        <p>Auteur : <?= $book['author'] ?></p>
                        <p>Année de publication : <?= $book['published_year'] ?></p>
                    </div>
                </div>
            </div>
        <?php endforeach; ?>
    </div>
</div>
```

Modification d'un élément



Afin de pouvoir modifier un élément, il va déjà nous falloir être en mesure de récupérer l'élément que l'on souhaite modifier, par conséquent, nous allons faire une méthode dans le model pour récupérer un livre à partir de son Id.

Nous serons en mesure d'avoir l'id du livre grâce à notre système de routage, il sera passer en paramètre en l'URL.

models/Book.php

```
1 usage

public static function getBookById($bookId){
    try {
        $db = connection();
        $stmt = $db->prepare( query: "SELECT * FROM books WHERE id = :id");
        $stmt->bindValue( param: ':id', $bookId, type: PDO::PARAM_INT);
        $stmt->execute();
        $book = $stmt->fetch( mode: PDO::FETCH_ASSOC);
        return $book;
    } catch (PDOException $e) {
        die('Erreur de requête : ' . $e->getMessage());
    }
}
```

controller/BookController.php

```
no usages

function update($id)
{
    $book = Book::getBookById($id);
    if (!$book) {
        $message = "Livre introuvable";
    } else {
        require_once('views/book/updateForm.php');
    }
}
```

Modification d'un élément



Pour vérifier que nous avons bien récupérer les informations de notre livre, on va commencer à remplir notre formulaire avec ses données. Pour ce faire, nous allons remplir les value de nos input comme ci-contre:

```
<div class="form-group">
    <label for="bookTitle">Titre du livre:</label>
    <input type="text" class="form-control"
        name="bookTitle" id="bookTitle"
        value=<?=$book['title']?>" maxlength="150">
</div>
```

models/Book.php

```
usage
public function update($id, $title, $author, $published_year) {
    try {
        $db = connection();
        $stmt = $db->prepare('UPDATE books
            SET title = :title, author = :author, published_year = :published_year, updated_at = NOW()
            WHERE id = :id');
        $stmt->bindValue( param: ':id', $id, type: PDO::PARAM_INT);
        $stmt->bindValue( param: ':title', $title, type: PDO::PARAM_STR);
        $stmt->bindValue( param: ':author', $author, type: PDO::PARAM_STR);
        $stmt->bindValue( param: ':published_year', $published_year, type: PDO::PARAM_INT);
        $stmt->execute();
    } catch (PDOException $e) {
        die('Erreur de requête : ' . $e->getMessage());
    }
}
```

Maintenant nous allons créer une méthode "update" pour notre modèle et l'appeler depuis notre contrôleur.

Suppression d'un élément



C'est le même principe pour la suppression, on crée la méthode dans le modèle, on crée l'action dans le contrôleur et on l'appelle dans la vue.

models/Book.php

```
1 usage
public static function delete($id) {
    try {
        $db = connection();
        $stmt = $db->prepare( query: 'DELETE FROM books WHERE id = :id' );
        $stmt->bindValue( param: ':id', $id, type: PDO::PARAM_INT );
        $stmt->execute();
        return true; // La suppression a réussi
    } catch (PDOException $e) {
        die('Erreur de requête : ' . $e->getMessage());
        return false; // La suppression a échoué
    }
}
```

views/book/library.php

```
<a href="delete/<?= $book['id'] ?>" class="btn btn-danger">Supprimer</a>
```

controller/BookController.php

```
no usages
function delete($id) {
    $book = Book::getBookById($id);
    if(!$book){
        $message = "Livre introuvable";
    } else {
        Book::delete($id);
    }

    $books = Book::getBooks();
    require_once ('views/book/library.php');
}
```

Soft Delete



La notion de "soft delete" (suppression douce) fait référence à une méthode de gestion des enregistrements dans une base de données où au lieu de supprimer physiquement un enregistrement, on marque l'enregistrement comme "supprimé" en mettant à jour un champ spécial (par exemple, `deleted_at`) avec un Datetime indiquant quand l'enregistrement a été supprimé.

Authentification



Afin de pouvoir nous authentifier sur notre application, nous allons commencer par créer un modèle "User.php" en respectant les normes d'encapsulation.

Pour débuter, les propriétés de notre classe seront:

models/User.php

```
1 usage
public static function create($username, $email, $password) {
    try {
        $db = connection();

        $stmt = $db->prepare('INSERT INTO users (username, email, password, created_at, updated_at)
            VALUES (:username, :email, :password, NOW(), NOW())');

        $hashedPassword = password_hash($password, [algo: PASSWORD_ARGON2ID ]);

        $stmt->bindValue([param: ':username', $username, type: PDO::PARAM_STR];
        $stmt->bindValue([param: ':email', $email, type: PDO::PARAM_STR];
        $stmt->bindValue([param: ':password', $hashedPassword, type: PDO::PARAM_STR]);
        $stmt->execute();

        $user_id = $db->lastInsertId();

        return new User($user_id, $username, $email, $hashedPassword, new DateTime(), new DateTime());
    } catch (PDOException $e) {
        die('Erreur de requête : ' . $e->getMessage());
    }
}
```

→ id	Int (11)
→ username	Varchar (255)
→ email	Varchar (255)
→ password	Varchar (255)
→ created_at	Datetime
→ updated_at	Datetime

Une fois notre modèle créé, nous allons pouvoir commencer l'inscription d'un utilisateur. Dans un premier temps on va faire la méthode create de notre modèle.

Formulaire d'inscription



Pour gérer toutes les actions (comme l'inscription ou la connexion) concernant un utilisateur, Il va nous falloir un nouveau contrôleur: "**UserController**".

Récupération des valeurs de mon formulaire



Vérification qu'il n'existe pas déjà un utilisateur avec cette adresse Email avec une méthode statique du modèle "User"



Création du nouvelle utilisateur



NB: N'oubliez pas de créer le formulaire (ici "signin.php") appelant cette action.

controller/UserController.php

```
no usages
function signin(){
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {

        $username = $_POST['username'];
        $email = $_POST['email'];
        $password = $_POST['password'];

        $existingUser = User::getByEmail($email);

        if ($existingUser) {
            $mailTaken = "L'adresse e-mail existe déjà.";
        } else {
            $user = User::create($username, $email, $password);

            if ($user) {
                $message = "Inscription réussie !";
            } else {
                $message = "Erreur lors de l'inscription.";
            }
        }
    }

    require_once 'views/signin.php';
}
```

La connexion



Dans cet exemple, nous allons gérer l'authentification de nos utilisateurs avec la session.

controller/UserController.php

```
no usages

function login(){
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $email = $_POST['email'];
        $password = $_POST['password'];

        $user = User::getByEmail($email);

        if ($user && password_verify($password, $user->getPassword())) {

            session_start();
            $_SESSION['user_id'] = $user->getId();
            $_SESSION['username'] = $user->getUsername();
            $_SESSION['email'] = $user->getEmail();
            header('Location:/mvc/Book/library');
            exit();
        } else {
            $message = "Identifiants incorrects. Veuillez réessayer.";
        }
    }

    require_once 'views/login.php';
}
```

views/login.php

```
<div class="container">
    <div class="panel panel-default">
        <div class="panel-heading">
            <h2 class="panel-title text-center">Connexion</h2>
        </div>
        <div class="panel-body text-center">
            <form action="login" method="post">
                <div class="row">
                    <div class="col-md-6 col-md-offset-3">
                        <div class="form-group">
                            <label for="email">Adresse e-mail</label>
                            <input type="email" class="form-control" id="email" name="email" required>
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-6 col-md-offset-3">
                        <div class="form-group">
                            <label for="password">Mot de passe</label>
                            <input type="password" class="form-control" id="password" name="password" required>
                        </div>
                        <p><?php if(isset($message)){echo ($message);}?></p>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-12 text-center">
                        <button type="submit" class="btn btn-primary">Se connecter</button>
                    </div>
                </div>
            </form>
        </div>
    </div>
</div>
```

Faire une application MVC avec Héritage

Objectif : Cet exercice a pour but de vous familiariser avec le concept d'héritage en PHP en créant une application simple de gestion de médiathèque en utilisant le Design Pattern MVC. Vous allez modéliser des objets représentant des médias (Livres, DVD, CD) qui peuvent être loués.

Créez une classe Media qui représente un média générique dans la médiathèque.

- titre (string) :
- auteur (string) :
- disponible (bool) :

Méthodes communes :

- emprunter()
- rendre()

Créer une classe enfant: Book avec la propriété pageNumber (Int)

Créer une classe enfant: Movie avec la propriété duration (double), et genre (enum)

Créer une classe enfant: Album avec la propriété songNumber (Int) et Editor (String)

Gestion de la Liste de médias:

- Être capable d'ajouter, modifier et supprimer des Médias, de les emprunter & de les rendre.
 - Uniquement pour les utilisateurs authentifiés
- Un utilisateur peut rechercher un film de manière approximative.
- L'utilisateur doit pouvoir tirer
- Les classes doivent être documentées.
- Gérer des illustrations par Médias en blob (Optionnel)

Faire une application MVC avec Héritage

Authentification :

- Mettez en place un système d'inscription et de connexion pour les utilisateurs.
- Utilisez des mots de passe hashés et des sessions pour gérer l'authentification.
- Le mot de passe doit être soumis à une regex.
 - Règle classique : 8 caractères min, majuscule, minuscule, chiffre et caractères spéciaux.
 - Le mot de passe ne doit pas contenir l'identifiant de l'utilisateur.
-

Tableau de Bord :

- Affichez un tableau de bord qui affiche la liste des médias et affiché si ils sont disponibles ou non
- Affichez le nom de l'utilisateur et un lien pour se déconnecter dans une barre de navigation.

Merci à vous !

Si vous avez des questions, je suis
joignable à l'adresse Email
ci-dessous.

Par Charles Haller
Email: charleshaller60@gmail.com