

Anthony Laplane



Introduction

PHP est un langage de programmation tout comme javascript à la différence que javascript est un langage client (lorsqu'il est utilisé sur une page web) alors que php est un langage serveur.

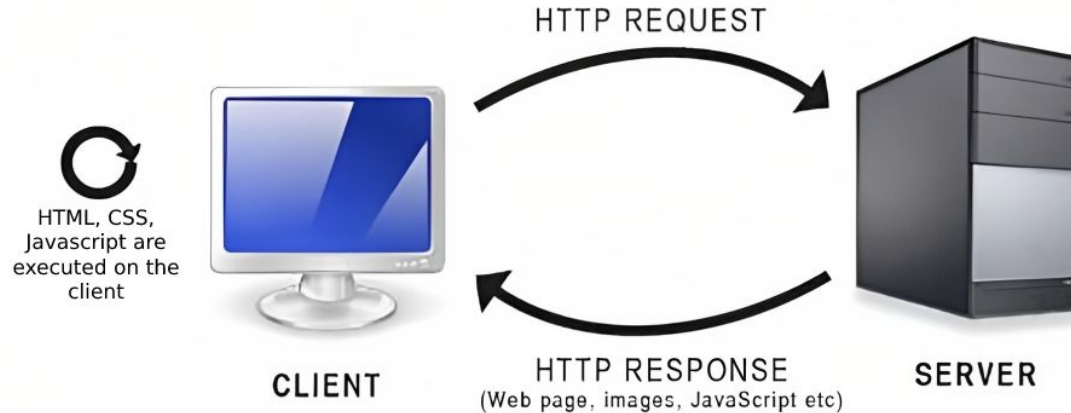
Langage exécuté côté client : Lorsque le code est exécuté, cela se passe sur l'ordinateur de l'utilisateur (le client). C'est le cas du html, css et javascript.

Langage exécuté côté serveur : Lorsque le code est exécuté, cela se passe sur le serveur.



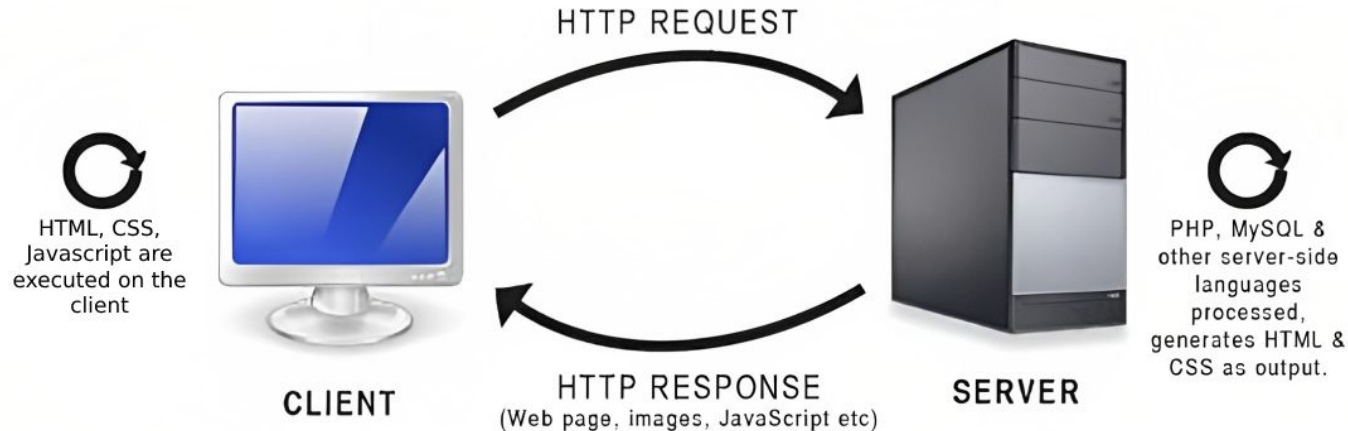
Architecture Client Server

- Le client demande une page web
- Le serveur prend en charge la demande
 - Dans le cas d'un site statique, il retourne directement le fichier html



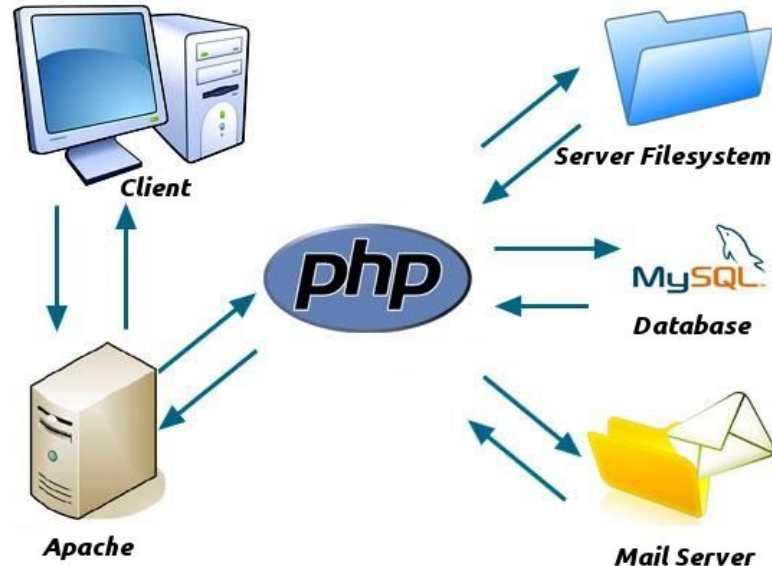
Architecture Client Server

- Dans le cas d'un site dynamique avec un langage backend
 - Le serveur va exécuter le langage backend (ex: php). On dit que le langage est exécuté côté serveur
 - Le html est généré dynamiquement (avec potentiellement une base de données)
 - Le serveur retour le contenu



Architecture Client Server

PHP est installé sur un serveur web (ex: Apache). PHP peut communiquer avec d'autres serveurs comme par exemple une base de données, un serveur d'email etc.



Introduction

Le langage php permet de créer des sites dynamiques (côté serveur) avec par exemple des données issues d'une base de données (ex: mysql, mariadb etc.).

La plupart des CMS utilisent le langage PHP (Wordpress, Drupal, Prestashop, Joomla etc.).

Environ 72% des sites utilisent PHP.

Quelques alternatives à PHP :



Environnement

1. Serveur local

- Mac: Install Mamp
<https://www.mamp.info/en/downloads/>
- Windows: Wamp
<https://www.wampserver.com/>

2. Modifier les “variables d’environnement” windows

- Ajouter le chemin de php

Fichier php

Un fichier php se termine par l'extension php (ex: page.php).

Le code php que l'on va écrire dans un fichier php va être entre une balise d'ouverture **<?php** et de fermeture **?>**

```
<?php
```

```
//je peux écrire entre les balises le code php
```

```
?>
```



Fichier php

Un fichier php va souvent mélanger du php et du html. Le html doit être écrit en dehors des balises php


```
<h1>Titre en html</h1>
```

```
<?php
```

```
//je peux écrire entre les balises le code php
```

```
?>
```

```
<h2>Titre en html</h2>
```



Echo

L'instruction echo va nous permettre d'afficher un message sur la page

```
<?php  
echo 'Bonjour';  
// ou  
echo('Bonjour');  
?>
```

echo

L'instruction Echo nous permet d'écrire un message dans la page. Pour afficher une chaîne, nous devons utiliser des guillemets.

```
<?php
echo "Hello";
?>

<p>
    <?php echo "Hello"; ?>
</p>

<p>
    <?="Hello"; ?>
</p>
```

Commentaires

Comme en javascript, on peut ajouter des commentaires dans notre code pour le rendre plus lisible.

```
<?php
// Commentaire sur une ligne

/*
  Un commentaire qui pourra
  contenir plusieurs ligne
*/
?>
```

Variables

Une variable est une zone mémoire qui pourra contenir une valeur. C'est une sorte de boîte dans laquelle on va pouvoir stocker une information.

```
<?php
// On déclare une variable age avec comme valeur 16
$age = 16;

// On affiche le contenu de la variable age sur la page avec echo
echo $age;

// Pour stocker une chaîne de caractère on utilise les guillemets simples ou double
$first_name = "John";
echo $first_name;

?>
```

Variables

Différence entre guillemets simples et doubles

```
<?php
$age = 16;

// En utilisant les guillemets doubles, php va pouvoir afficher le contenu de la variable
echo "I am $age years old ";

// En utilisant les guillemets simples, il faudra concaténer en utilisant le point
echo 'I am '.$age.' years old';
?>
```

Exercice 1

- Dans le dossier cours_php, créez un nouveau fichier exercice1.php
 - En haut du fichier, ajoutez des balises php et ajoutez quelques variables
 - first_name
 - last_name
 - age

Afficher l'affichage avec echo dans un paragraphe:

Je m'appelle XX XX . J'ai XX ans.

Exercice 2

- Dans VSCODE, créez un nouveau fichier exercice2.php (dans php_cours)
- Ajouter la structure HTML
- En haut du fichier, ajoutez des balises php et ajoutez quelques variables
 - `product_name`
 - `description`
 - `price`
- Dans le body afficher le `product_name` dans un `h1`, la `description` dans un paragraphe et le `price` dans un `h2`

Types de variables

- **int** : un nombre entier, positif ou négatif.
- **float** : Un nombre décimal.
- **string**: une chaîne de caractères, soit entre guillemets simples, soit entre guillemets doubles.
- **bool** : une valeur qui peut être true ou false.
- **array** : une collection de valeurs, chacune accessible par un index numérique ou une clé.
- **object** : une instance d'une classe définie par l'utilisateur.
- **null** : une valeur spéciale qui représente une absence de valeur.



Constantes

En PHP, une constante est une valeur qui ne peut pas être modifiée pendant l'exécution du script. Une constante est définie grâce à la fonction "define()", qui prend deux arguments : le nom de la constante et sa valeur.

```
<?php

// Une constante n'est défini qu'une seule fois
define("MAJORITY", 18);

echo MAJORITY;

?>
```

Instruction if

Les conditions en PHP sont définies grâce à des mots-clés tels que "if" (si), "else" (sinon) et "elseif" (sinon si).

```
<?php
// On déclare une variable age avec comme valeur 16
$age = 16;

// On teste l'âge. Si l'âge est supérieur à 18, on affiche un message, sinon un
autre message
if ($age >= 18) {
    echo 'adulte';
} else if ($age >= 13) {
    echo 'ado';
} else {
    echo 'enfant';
}
?>
```

Echo suite

L'instruction echo va nous permettre d'afficher un message mais aussi des variables avec guillemet simple ou double.

La différence étant qu'avec guillemet double, php affichera le contenu de la variable alors que ce n'est pas le cas avec guillemet simple :

```
<?php
$age = 30;

echo "age : $age ans"; // Affichage -> age : 30 ans
echo 'age : $age ans'; // Affichage -> age : $age ans

// Un raccourci pour echo intéressant quand on mélange avec du html :
?>
<p>Vous avez <?=$age; ?> ans</p>
```


Exercice 3

- Dans VSCODE, créer un nouveau fichier exercice3.php
- Dans ce fichier ajouter la structure html
- En haut du fichier, dans des balises php ajouter les variable suivantes avec les valeurs de votre choix
 - user_name
 - age
- Dans le body
 - Si l'age est inférieur à 18
 - Afficher dans un paragraphe la phrase suivante "Notre jeu est accessible aux personnes de plus de 18 ans"
 - Sinon afficher
 - "Bienvenue sur notre jeu XXX" (remplacer XXX par le nom d'utilisateur)

Exercice 3b

- Dans VSCODE, créez un nouveau fichier exercise3b.php avec une structure HTML
- En haut du fichier, ajoutez des balises php et ajoutez quelques variables
 - `gameName`
 - `isNew (bool)`
- Dans le body
 - Afficher le nom du jeu dans un paragraphe
 - Si `isNew` est vrai, affichez « new » en gras à l'intérieur du p.
Exemple : Little Nightmares 3 - **NEW!**
 - Sinon on affiche uniquement le nom du jeu
Exemple : Little Nightmares 3

Les opérateurs

- Opérateurs de comparaison de valeur :
 - ">" (strictement supérieur à)
 - "<" (strictement inférieur à)
 - ">=" (supérieur ou égal à)
 - "<=" (inférieur ou égal à)
 - "==" (égal à) ou "===" (pour vérifier le type)
 - "!=" (différent de) ou "!==" (pour vérifier le type)
 - Opérateurs logiques en PHP :
 - "&&" (ET logique)
 - "||" (OU logique)
 - Opérateurs arithmétiques :
 - "+" (addition)
 - "-" (soustraction)
 - "*" (multiplication)
 - "/" (division)
 - "%" (modulo)
- 

Instruction switch

switch permet de comparer une expression à une liste de valeurs. Elle peut être utilisée de manière similaire à la structure "if"/"elseif"/"else", mais peut être plus concise et plus lisible dans certains cas.

```
<?php
$numJour = 3;

switch ($numJour) {
    case 1:
        echo "Lundi";
        break;
    case 2:
        echo "Mardi";
        break;
    case 3:
        echo "Mercredi";
        break;
    default:
        echo "Numéro de jour invalide";
}
?>
```



php 8.0 a introduit l'instruction match

<https://dev.to/mainick/php-match-expression-match-vs-switch-3j5b>

Les tableaux

Un tableau (array en anglais) est une structure de données qui permet de stocker une liste d'éléments de manière organisée.

```
<?php
// Tableau simple
$fruits = ["Banane", "Orange", "Pomme"];
echo $fruits[0]; // On accède à un élément du tableau par son index/clé, affiche "Banane"
```

Note : Il existe une syntaxe alternative :

```
$fruits = array("Banane", "Orange", "Pomme");
echo $fruits[0];
```

Les tableaux

Un tableau peut contenir des éléments de n'importe quel type (chaînes de caractères, nombres, objets, etc.). Chaque élément est composé d'une **clé/index** et de la **valeur**.

```
<?php
    // Tableau associatif
    $utilisateur = ["nom" => "Dupond", "prenom" => "Jean", "age" => 26];
    echo $utilisateur["nom"]; // Affiche "Dupond"
?>
```

Exercice 4

- Dans VSCODE, créer un nouveau fichier exercice4.php
- Ajouter la structure html
- Tout en haut du fichier, ajouter la balise php et un tableau associatif user (first_name, last_name, email)
- Dans le body, afficher les informations:
 - first_name, last_name dans un h1
 - email dans un paragraphe

Les tableaux multidimensionnels

Un tableau multidimensionnel est un tableau qui contient d'autres tableaux en tant qu'élément. Cela permet de créer des structures de données plus complexes pour stocker et manipuler des informations.

```
<?php
$utilisateurs = [
    ["nom" => "Dupond", "prenom" => "Jean", "age" => 26],
    ["nom" => "Martin", "prenom" => "Rose", "age" => 35],
    ["nom" => "Doe", "prenom" => "Jane", "age" => 31]
];

?>
```

Boucle foreach

La boucle "foreach" permet de parcourir un tableau et d'exécuter un bloc de code pour chaque élément du tableau. La boucle "foreach" prend en compte les **clés** et les **valeurs** du tableau.

Exemple sans clé

```
<?php
    $fruits = ["Banane", "Orange", "Pomme"];

    foreach ($fruits as $key=>$value) {
        echo "<p>$value</p>";
    }
?>
```

Boucle foreach

Exemple avec tableau multidimensionnel

```
<?php
$utilisateurs = [
    ["nom" => "Dupond", "prenom" => "Jean", "age" => 26],
    ["nom" => "Martin", "prenom" => "Rose", "age" => 35],
    ["nom" => "Doe", "prenom" => "Jane", "age" => 31]
];

foreach ($utilisateurs as $index=>$utilisateur) {
    echo "<h2>Nom : " . $utilisateur["nom"] . "</h2>";
    echo "<p>Prénom : " . $utilisateur["prenom"] . "</p>";
    echo "<p>Age : " . $utilisateur["age"] . "</p>";
}
?>
```

Exercice 5

- Créer un nouveau fichier exercice5.php avec la structure html
- Tout en haut, créer un tableau (array) multidimensionnel de **votre choix**.
- Afficher les données avec un foreach dans le body avec une structure ul/li
- Quelques exemples :
 - Tableau de films (avec titre, année, réalisateur)
 - Tableau produits (avec nom, prix, catégorie)
 - etc.

Atelier 1

- Créez un fichier atelier1.php et à l'intérieur :
- Ajouter la structure HTML
- En haut du fichier, ajoutez un tableau \$countries et ajouter plusieurs pays avec nom et population (tableau associatif)
- Dans le body
 - Ajouter un h1 "Atelier 1"
 - Ajouter un h2 "Liste des pays"
 - utilisez un foreach pour afficher le nom du pays et sa population dans une liste HTML (ul/li) ou dans un tableau
- Dans le foreach, ajouter un if pour stocker le pays le plus peuplé et l'afficher en dessous du ul/li dans un h2 sous la forme:
 - <h2>Le pays le plus peuplé est XX avec une population YY</h2>

Boucle for

La boucle "for" permet d'exécuter un bloc de code plusieurs.

La boucle "for" se compose de trois parties : l'initialisation, la condition de fin de boucle et l'incrémentation.

```
<?php
    for ($i = 1; $i <= 10; $i++) {
        echo "$i ";
    }
    // Affiche : 1 2 3 4 5 6 7 8 9 10
?>
```

Boucle while

La boucle "while" permet d'exécuter un bloc de code de manière itérative tant qu'une condition est vraie. Elle se compose d'une condition qui est évaluée avant chaque itération de la boucle.

Exemple sans clé

```
<?php
    $i = 1;
    while ($i <= 10) {
        echo "$i ";
        $i++;
    }
    // Affiche : 1 2 3 4 5 6 7 8 9 10
?>
```

Les fonctions

Une fonction est un bloc de code qui peut être exécuté de manière autonome et qui peut éventuellement retourner une valeur. Une fonction peut être appelée à plusieurs reprises dans un programme, ce qui permet de réutiliser du code et de rendre celui-ci plus lisible et maintenable.

```
<?php
function carre(float $nombre):float
{
    return $nombre * $nombre;
}

echo carre(5); // Affiche 25

?>
```



On peut typer les arguments et le retour.

Même si ce n'est pas obligatoire, c'est fortement conseillé pour rendre son code plus robuste.

Exercice 6

- In VSCODE, créer un nouveau fichier exercice6.php
- Ajouter la structure html
- Tout en haut du fichier, ajouter une balise php pour ajouter du code php
 - Créer une fonction surface (ex: calculateArea) qui prend deux arguments longueur, largeur et qui retourne le produit des deux afin de calculer la surface
- Dans le body, appeler cette fonction pour afficher la surface de différentes pièce et gérer l'affichage :
 - Chambre : 9m²
 - Salon : 20m²

Atelier 2

- Créez un fichier atelier2.php
- Ajouter la structure HTML
 - En haut du fichier
- ajouter un tableau avec 3 jeux (nom et age minimum) exemple:
 - ["name" => "GTA V", "minimumAge"=> 18]
- Toujours en haut, ajoutez une fonction isAllowedToPlay()
 - Cette fonction prendra deux arguments \$gameAge et \$userAge
- Cette fonction comparera les deux âges et renverra vrai ou faux
- Toujours en haut du fichier, créez une variable \$age = 7;
- Dans le body
 - Faire un foreach sur le tableau et **appelez isAllowedToPlay() pour chaque jeu** en passant deux paramètres pour vérifier si l'utilisateur est autorisé
 - Cela affichera pour chaque jeu:
 - Vous êtes autorisé à jouer à XX
 - Vous n'êtes pas autorisé à jouer à YY



include vs require ?

Include et require

Les instructions "include" et "require" permettent d'inclure et d'exécuter un fichier au sein d'un autre fichier. Elles permettent de partager du code commun entre plusieurs fichiers et de rendre celui-ci plus lisible et maintenable.

```
<?php  
include 'fichier1.php';  
require 'fichier2.php';  
?>
```

La différence entre ces deux instructions est que require va générer une exception si le fichier est introuvable alors qu'include affichera seulement un warning.



Include_once et require_once

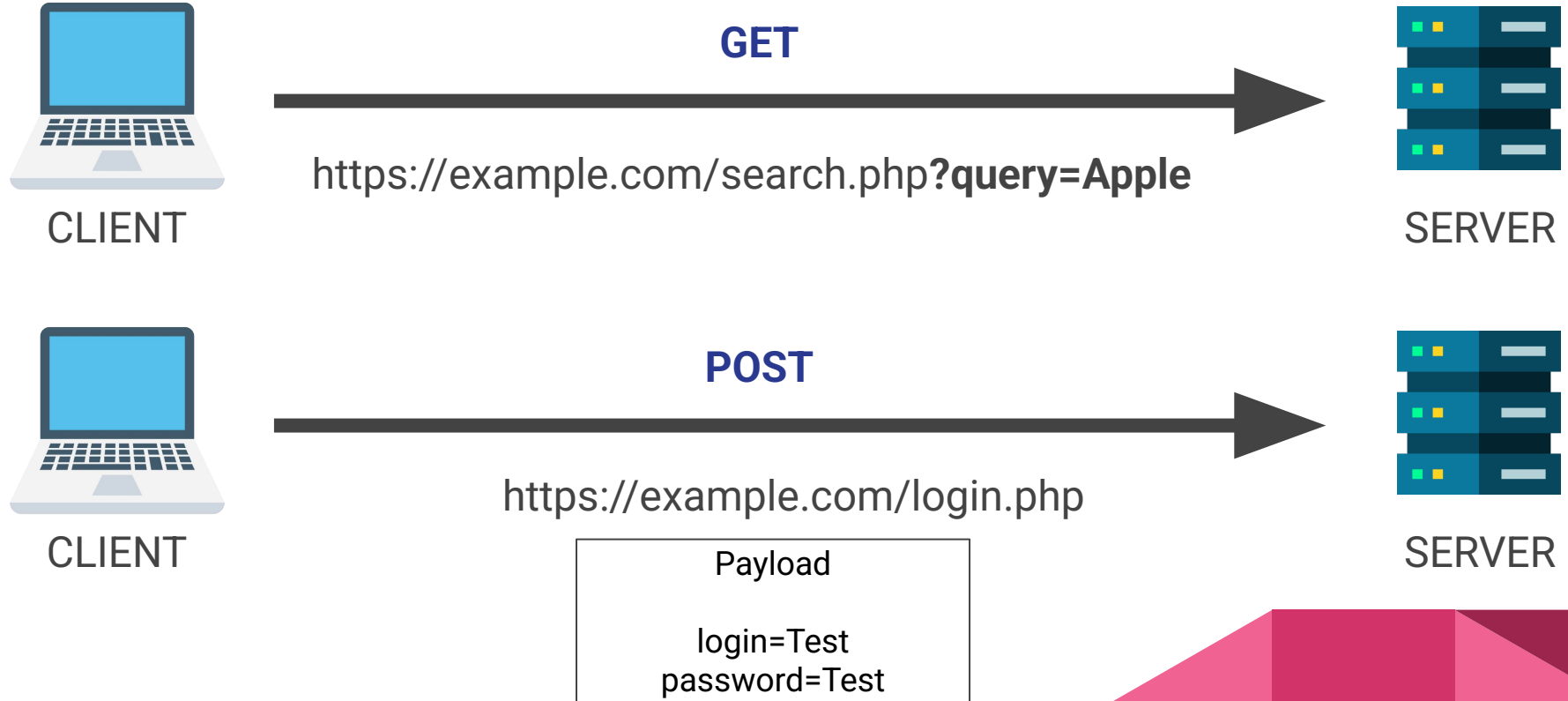
"include_once" et "require_once" fonctionnent de manière similaire à "include" et "require", mais avec une différence importante : si le fichier inclus a déjà été inclus dans le script en cours d'exécution, il ne sera pas inclus de nouveau.

```
<?php  
include_once 'fichier1.php';  
require_once 'fichier2.php';  
?>
```



GET et POST

Les méthodes GET vs POST



\$_GET

\$_GET est un tableau associatif qui contient les données envoyées au serveur via la méthode "GET" d'un formulaire HTML ou d'une URL.

Les données passées par une adresse url sont de la forme suivante :

mapage.php?nom=dupond&prenom=jean

On pourra ensuite y accéder en php de la manière suivante :

```
<?php
echo $_GET['prenom']; // ATTENTION : va générer un warning
if (isset($_GET['nom'])) {
    echo htmlspecialchars($_GET['nom']);
}
?>
```

Prévenir les failles XSS (cross-site scripting)

Utiliser htmlspecialchars() avant d'afficher des données

```
<?php
$dataFromDatabase = "<script>alert('XSS attack!');</script>";
echo htmlspecialchars($dataFromDatabase);
```

Si on souhaite autoriser certaines balises, on peut utiliser cette librairie:

<http://htmlpurifier.org/>



Exercice 9

- Dans VSCODE, créer un nouveau fichier exercice9.php
- Ajouter les structure html
- Si un paramètre “nom” est présent dans l'url, le récupérer et afficher le message suivant :
 - Joyeux anniversaire XXX
 - sinon afficher “le paramètre nom n'est pas présent dans l'url”

\$_POST

\$_POST est un tableau associatif qui contient les données envoyées au serveur via la méthode "POST" d'un formulaire HTML.

On pourra ensuite y accéder en php de la manière suivante:

```
<?php
if (isset($_POST['nom'])) {
    echo $_POST['nom'];
}
?>
```

Exercice 10

- In VSCODE, créer un fichier exercice10.php
- Ajouter la structure html
- Ajouter un formulaire les champs
 - input text first_name
 - input text last_name
 - input submit
- Le formulaire doit avoir la methode POST
- Une fois les données envoyé, afficher le nom et prénom de l'utilisateur
- Exemple d'affichage: Bonjour John Doe

Atelier 2 suite

- Continuer dans votre fichier atelier2
- On ne veut plus utiliser la variable \$age qui était en dur
- Ajouter un formulaire post qui demande l'âge de l'utilisateur
- Comme précédemment, afficher la liste des jeux mais cette fois basée sur l'âge réel de l'utilisateur
- Quand la page est chargée la première fois sans l'âge, afficher un message:
 - Saisir votre age pour savoir à quels jeux vous pouvez jouer

Exercice couleur

- Créer un nouveau fichier php `exercice_couleur.php`
- Créer un formulaire qui demande à l'utilisateur de rentrer sa couleur préférée :
 - input type text avec en name color
 - input type submit
- Utiliser la méthode POST et laisser vide en action
- Lors de la soumission du formulaire afficher un message
 - "Votre couleur préférée est XXX"
- Quand tout fonction, ajouter un if:
 - Si la couleur est rouge, alors afficher le message suivant:
 - "rouge est aussi ma couleur préférée !"

TP Météo

- Le but est de faire une liste déroulante avec des villes. Quand l'utilisateur sélectionne une ville, on affiche les données météo.
- Créer un dossier **tp_meteo**
- Ajoutez un index.php avec la structure html
- en haut du fichier ajouter un tableau (array) multidimensionnel avec plusieurs villes et des données météo:
 - ["name" => "Montpellier", "weather" => "ensoleillé", "min_temp" => 1, "max_temp"=>6]
- Dans le body, ajouter un formulaire avec un select permettant de choisir la ville
 - il faudra faire un foreach dans le select pour afficher tous les éléments <option> et mettre en value l'index
- Si on reçoit en post la ville sélectionnée, il faut alors afficher les informations de cette ville:
<h2>Montpellier</h2>
<h3>Pluie</h3>
<p>Température mini: 12° - Température max: 22°</p>

Atelier 3 (moviz)

- Faire un fork et cloner ce projet:
 - https://github.com/arirangz/moviz_template
- Créer un dossier nommé **libs**
- Dans le dossier libs, créez un fichier movie.php et à l'intérieur
 - Créez une fonction getMovies() et dans la fonction créer un tableau avec 3 films (title, summary, release_date) et retourner ce tableau
 - Créez une fonction getMovieByIndex(\$index) qui prend un argument \$index et renverra le film. Cette fonction va appeler getMovies()
- Dans index.php, appelez getMovies() et faire un foreach pour afficher uniquement le nom de chaque film et un lien “En savoir plus” qui mène vers movie_details.php en ajoutant l'id en paramètre.
- Ajoutez un fichier movie_details.php à la racine (avec header et footer)
- Dans movie_detail.php, vous devez appeler getMovieByIndex et afficher toutes les informations sur ce film

Atelier 3 partie 2 (moviz)

- Continuer dans moviz_template
- Dans ce dossier créez une page login.php (importer le header et footer)
- Récupérer un formulaire de connexion bootstrap
- La méthode du formulaire doit être post et action vide
- Vérifiez les données \$_POST (avec isset)
 - si l'e-mail est "test@gmail.com" et le mot de passe est "ABC123", alors affichez
 - Accès autorisé
 - sinon afficher
 - Accès refusé
- Lorsque le code fonctionne, déplacez le if dans une fonction loginUser() (dans un fichier libs/user.php). Cette fonction attend deux param (email, password) et retourne un bool



On ne mettra jamais les données de connexion en dur sur un site réel

LES SESSIONS

Les sessions

Une session est une manière de stocker des données côté serveur pour un utilisateur spécifique pendant sa visite sur un site web. Elle permet de conserver des informations entre différentes requêtes HTTP.

En php on devra tout d'abord démarrer une session avec `session_start()`.

On pourra ensuite manipuler les données de session avec le tableau `$_SESSION`.

```
// Démarre une nouvelle session
session_start();

// Stocke des informations dans la session
$_SESSION['username'] = 'johndoe';
```

Les sessions : connexion

- Connecter un utilisateur passe par deux phases :
 - On l'authentifie avec son login et mot de passe
 - On persiste (on stocke) les informations dans une session

```
session_start();

if ($_POST['username'] === "admin" && $_POST['password'] === "XD2Ka@" ) {
    //Prévient les attaques de fixation de session
    session_regenerate_id(true);
    $_SESSION['username'] = $_POST['username'];
}
```

Les sessions : logout

Lorsque l'on souhaite déconnecter un utilisateur, nous allons devoir supprimer les données de session avec `session_destroy()`. L'appel à cette fonction va détruire les données du serveur mais pas vider le tableau `$_SESSION`.

Nous allons donc devoir supprimer le tableau de session avec un `unset()`.

```
session_start();  
//Supprime les données de session du serveur  
session_destroy();  
//Supprime les données du tableau $_SESSION  
session_unset();
```

Régénérer l'ID de session

Régénérer l'ID de session après la connexion et la déconnexion : Après une connexion réussie ou une déconnexion, il est recommandé de régénérer l'ID de session en utilisant la fonction `session_regenerate_id`. Cela empêche les attaques de fixation de session, où un attaquant peut fixer l'ID de session avant que l'utilisateur ne se connecte.

```
<?php  
session_regenerate_id(true);
```

<http://www.web-d.be/post/94/l'attaque-par-fixation-de-session,-et-comment-s'en-prot%C3%A9ger.html>



Protéger le cookie de session

secure (ne fonctionnera pas en local sans https)

paramètre des cookies qui indique au navigateur que le cookie doit être envoyé uniquement via une connexion HTTPS sécurisée. Cela signifie que le cookie ne sera pas transmis sur une connexion HTTP non sécurisée.

httponly

paramètre des cookies qui restreint l'accès aux cookies via JavaScript. Lorsque l'attribut "HttpOnly" est défini sur true, le cookie ne peut être accédé ou modifié que par le serveur via des requêtes HTTP.

```
<?php
session_set_cookie_params([
    'lifetime' => 3600,
    'path' => '/',
    'domain' => 'example.com',
    'secure' => true,
    'httponly' => true
]);
session_start();
```

Protéger le cookie de session

Secure

Paramètre des cookies qui indique au navigateur que le cookie doit être envoyé uniquement via une connexion HTTPS sécurisée. Cela signifie que le cookie ne sera pas transmis sur une connexion HTTP non sécurisée.

<https://php.net/session.cookie-secure>

SameSite

Paramètre des cookies qui contrôle le comportement de transmission des cookies en fonction de l'origine de la requête. Il permet de limiter les risques de certaines attaques, telles que les attaques de type Cross-Site Request Forgery (CSRF).

<https://tools.ietf.org/html/draft-west-first-party-cookies-07>

StrictSession

Lorsque le mode strict des sessions est activé, les cookies de session sont associés uniquement à l'adresse IP du client qui les a créés. Cela signifie que si l'adresse IP du client change entre les requêtes, la session sera invalidée.

https://wiki.php.net/rfc/strict_sessions

UseCookie et UseOnlyCookie

Permet de forcer l'utilisation de cookie pour la session (au lieu de le passer en get ou post)

<https://php.net/session.use-cookies>



Atelier3 partie 3 (moviz)

- Continuer dans le dossier moviz_template
- Si l'utilisateur et le mot de passe sont corrects, on souhaite stocker l'email de l'utilisateur dans la session.
- En haut de votre page, afficher l'email de l'utilisateur connecté et un lien vers une page logout.php
- Créez ensuite une page logout.php pour détruire la session et rediriger vers login.php

Atelier3 partie 4 (moviz)

- Dans profile.php ajoutez un h1 « Mon profil » et un h2 affichant l'email de l'utilisateur
- Si l'utilisateur tente d'accéder à profile.php et n'est pas connecté, il sera redirigé vers login.php
- Modifiez login.php pour rediriger l'utilisateur sur la page profile.php si l'email et le mot de passe sont correctes
- Dans le menu:
- Si l'utilisateur est connecté il faut afficher déconnexion et profil sinon login

ACCÈS AUX DONNÉES AVEC PDO

PDO

PDO (PHP Data Objects) est une extension de PHP. Elle permet de gérer les bases de données de manière uniforme, quel que soit le type de base de données utilisé (MySQL, SQLite, etc.). Avec PDO nous allons pouvoir nous connecter à une base de données, exécuter des requêtes SQL et récupérer les résultats. PDO offre également des fonctionnalités de sécurité pour protéger contre les injections SQL.

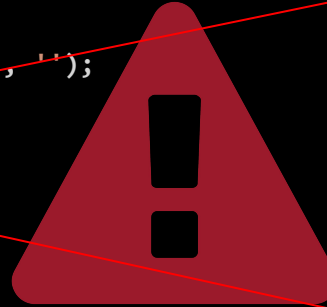


PDO



Ci-dessous un exemple d'une requête **NON SÉCURISÉ**

```
<?php
$pdo = new PDO('mysql:dbname=my_db;host=localhost;charset=utf8mb4', 'root', '');
$id = $_GET['id'];
$query = $pdo->query("SELECT * FROM user WHERE id = $id");
$user = $query->fetch(PDO::FETCH_ASSOC);
?>
```



Ce code n'est pas sécurisé car un attaquant pourra injecter du code dans le paramètre d'url :

?id=5;DELETE FROM users;

La requête suivante sera alors exécutée :

SELECT * FROM user WHERE id =5;DELETE FROM user;

PDO

PDO nous permet de préparer, exécuter et récupérer des données.

On commence par instancier un objet PDO dans un try/catch pour gérer les erreurs.

```
<?php
try
{
    $pdo = new PDO("mysql:dbname=my_db;host=localhost;charset=utf8mb4", "root", "");
}
catch (Exception $e)
{
    die('Erreur : ' . $e->getMessage());
}
?>
```

PDO

Avec l'objet PDO nous pouvons effectuer des requêtes **préparées**

La préparation de requête avec `bindValue()` nous permet de sécuriser nos requêtes.

```
<?php
$id = (int)$_GET['id'];
$query = $pdo->prepare("SELECT * FROM user WHERE id = :id");
$query->bindValue(':id', $id, PDO::PARAM_INT);
$query->execute();
//fetch() nous permet de récupérer une seule ligne
$user = $query->fetch(PDO::FETCH_ASSOC);
//$user est un tableau association qu'on peut manipuler comme on l'a vu précédemment
?>
```

PDO

Pour récupérer plusieurs lignes, on utilise fetchAll

```
<?php
$query = $pdo->prepare("SELECT * FROM user");
$query->execute();
$users = $query->fetchAll(PDO::FETCH_ASSOC);
// $users sera un tableau contenant une ligne par utilisateur. Il faudra donc faire une
foreach pour parcourir ce tableau
?>
```

Atelier3 partie 5

- Dans phpmyadmin, créez une nouvelle base de données “moviz”
- Importez ensuite ce fichier SQL :
<https://drive.google.com/file/d/1SJtFS7vJy-zKEsRgYn3XTQcMUjKDvRJ4/view?usp=sharing>
- Ajouter un fichier pdo.php dans libs (qui contient la connexion à la bdd) et faire un require_once de ce fichier dans index.php et dans movie_details.php
- Modifier les fonctions pour récupérer les films depuis la base de données au lieu de tableau (il va falloir ajouter un paramètre \$pdo à chaque fonction)
- Sur l'index, vous devez afficher la liste des films avec uniquement le titre et un lien pour en savoir plus (remplacer l'index par l'id).
- En cliquant sur lire la suite, nous allons sur une page movie_details.php qui affichera toutes les informations du film

PDO - Mot de passe en bdd

Lorsqu'on va enregistrer le mot de passe utilisateur dans la base de données, il ne faut surtout pas le stocker en clair.

On va utiliser la fonction **password_hash** pour hasher le mot de passe (attention à ne pas confondre les termes hasher, chiffrer, ~~encrypter~~, encoder).

<https://www.php.net/manual/en/function.password-hash.php>

```
<?php
```

```
$query = $pdo->prepare("INSERT INTO user (email, password)
                        VALUES (:email, :password)");

$query->bindValue(':email', $email);
$hash = password_hash($password, PASSWORD_DEFAULT);
$query->bindValue(':password', $hash);
$result = $query->execute();
```

PDO - Mot de passe en bdd

Comme le mot de passe est hasher, lorsqu'un utilisateur va se connecter au site, on ne pourra pas directement comparer le mot de passe de l'utilisateur avec celui en base de donnée.

On va devoir utiliser la fonction `password_verify()` qui retournera true si le mot de passe et le hachage correspondent.

```
<?php
    // On récupère au préalable l'utilisateur en bdd
    // ...

    password_verify($email, $password);
```

Atelier3 partie 6

- Modifier votre code dans login.php pour créer un véritable système de connexion en utilisant l'utilisateur dans la base de données (l'utilisateur est « test@test.com » avec le mot de passe « test »).
- Créer ensuite une nouvelle page signup.php avec un formulaire et les champs suivants
 - nickname
 - email
 - password
- Si tous les champs ne sont pas vides, enregistrer l'utilisateur dans la base de données et rediriger vers la page login.php

Atelier Techtrendz

- Suivre les instructions du README :
https://github.com/arirangz/techtrendz_template
- Finaliser le projet puis faire un commit et push



LA P00

(Programmation Orienté Objet)

Programmation Orienté Objet

La Programmation Orientée Objet (POO) est un paradigme (une manière de conceptualiser, d'organiser) de programmation qui utilise des "objets" et des "classes" pour organiser le code.

La POO facilite la modularité, la réutilisation du code et le travail en équipe, tout en permettant de créer des structures de code plus lisibles et maintenables.



Programmation Orienté Objet

Quelques principes que nous allons voir dans l'utilisation de la POO avec PHP :

- **Encapsulation** : Le regroupement de données et de fonctions qui manipulent ces données en un seul objet.
- **Héritage** : La capacité pour une classe d'hériter des propriétés et méthodes d'une autre classe.
- **Polymorphisme** : La capacité pour des objets de différentes classes de traiter des méthodes qui ont le même nom de manière différente.
- **Abstraction** : La capacité de créer des classes abstraites et des interfaces qui définissent des méthodes et des structures sans implémenter leur fonctionnalité complète.



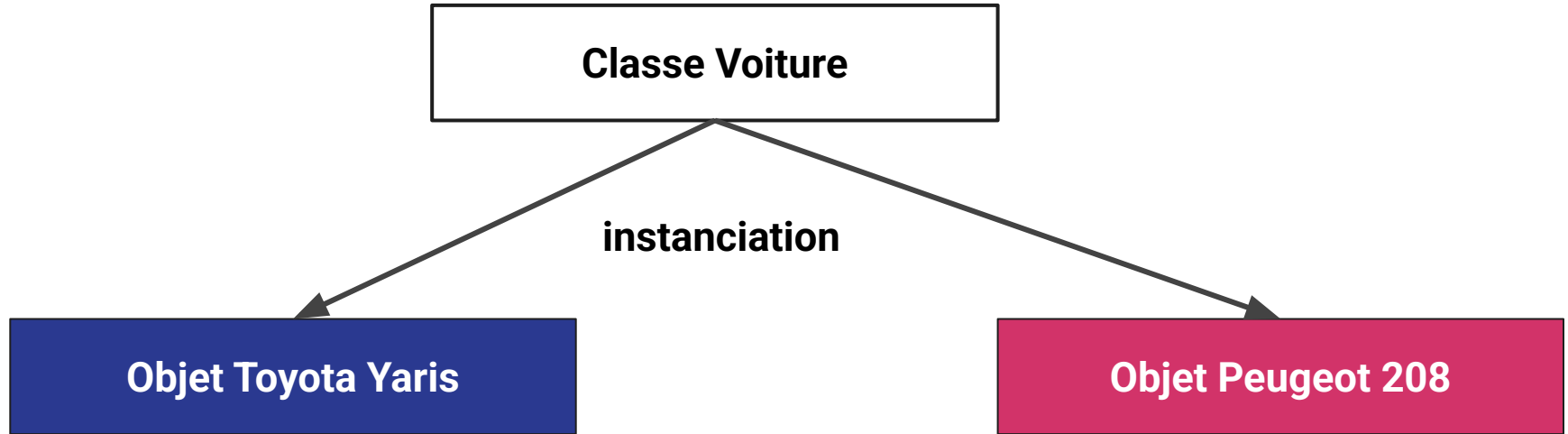
Programmation Orienté Objet

Deux concepts importants

- **Classe** : Une classe est une sorte de modèle ou de moule permettant de créer des objets. Le fait de créer un objet à partir d'une classe s'appelle l'**instanciation**. Exemple : une classe Voiture
- **Objet** : Un objet est créé à partir d'une classe. Chaque objet construit à partir d'une classe peut être différent. Exemple : un objet Toyota Yaris, un objet Peugeot 208.



Programmation Orienté Objet



Créer une classe

```
class Voiture
{
    public string $marque;
    public float $vitesseMax;
}
```

Pour créer une classe, on utilise le mot clé `class` suivi du nom de la classe en Pascal Case (ou UpperCamelCase) ex: `MaClasse`.

A l'intérieur de la classe, on va pouvoir définir des variables qu'on appelle des **propriétés**.

Pour chaque propriété, on va pouvoir définir son **type** (string, float, bool) et sa **visibilité** (public, protected, private) que nous verrons plus tard.

Le nom des propriétés est en général en lower camelCase (ex : `maPropriete`).



Créer un objet : l'instanciation

```
class Voiture
{
    public string $marque;
    public float $vitesseMax;
}

$voiture1 = new Voiture();
$voiture1->marque = "Toyota";
$voiture1->vitesseMax = 275;
```

On **instancie** une classe (le fait de créer un objet) en utilisant le mot clé new suivit du nom de la classe.

Une fois la classe instanciée, on obtient un objet qu'on peut manipuler.

Si l'on souhaite modifier les propriétés on peut utiliser l'objet suivi d'une flèche -> puis le nom de la propriété et écrire une valeur.



Ajouter des méthodes

```
class Voiture
{
    public string $marque;
    public float $vitesseMax;

    public function klaxonner():void
    {
        echo 'bip bip !';
    }
}

$voiture1 = new Voiture();
$voiture1->marque = "Toyota";
$voiture1->vitesseMax = 275;
$voiture1->klaxonner();
```

On peut ajouter des fonctions à une classe, on appelle cela des **méthodes**.

Ensuite on peut appeler cette méthode à partir de l'objet.



Le constructeur

```
class Voiture
{
    public string $marque;
    public float $vitesseMax;

    public function __construct(string $marque, float $vitesseMax)
    {
        $this->marque = $marque;
        $this->vitesseMax = $vitesseMax;
    }
}

$voiture1 = new Voiture("Toyota", 275);
```

Pour faciliter la création d'objet (instanciation), on peut passer les différents paramètres entre parenthèse.

La méthode **__construct** va alors être appelée automatiquement (c'est une méthode magique).

Dans cette méthode, on va utiliser **\$this** qui fait référence à l'objet courant.

Le constructeur

```
class Voiture
{
    public function __construct(
        public string $marque,
        public float $vitesseMax
    ) {
    }
}

$voiture1 = new Voiture("Toyota", 275);
```

Depuis la version 8 de php on peut utiliser la promotion des propriétés dans le constructeur.



Exercice 12

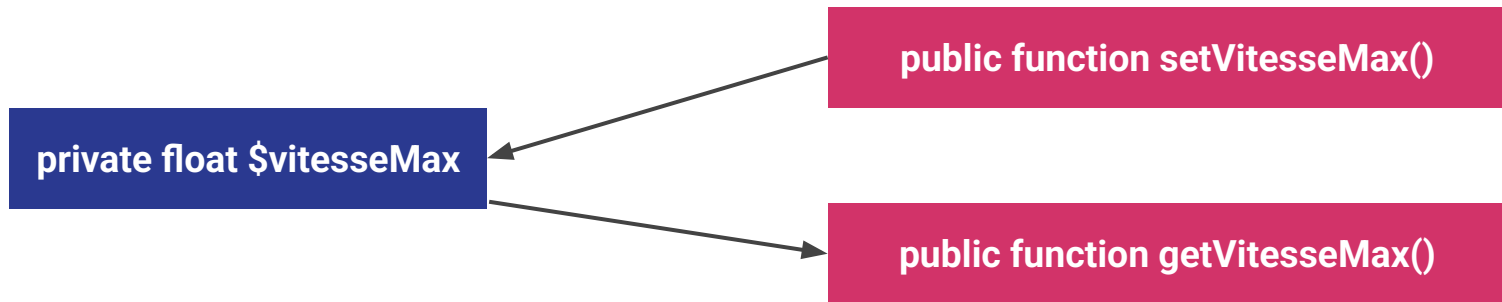
- Créer un nouveau dossier **exercice_formes**
- Créer un nouveau fichier Rectangle.php
- Dans ce fichier créer une classe du même nom avec :
 - propriété longueur
 - propriété largeur
 - propriété couleur
 - méthode calculerSurface() qui retourne le produit des deux
- Ajouter un fichier index.php qui importe la classe
- Dans le fichier index.php
 - Faire un require_once du fichier Rectangle.php
 - Faire un nouvel objet
 - Appeler la méthode calculerSurface() et afficher le résultat
 - Afficher la couleur

P00 avancé

L'encapsulation : les getters et setters

Le fait de pouvoir manipuler les propriétés directement à un inconvénient : on n'a pas de contrôle sur ce qui est ajouté.

Un des principes de l'encapsulation va être de rendre mettre les propriétés en “private” ou “protected”. Ceci aura pour effet de rendre ces propriétés non visibles en dehors de la classe. Pour les manipuler on utilisera des méthodes qu'on appelle les getters et les setters.



L'encapsulation : les getters et setters

```
class Voiture
{
    private string $marque;
    private float $vitesseMax;

    public function setVitesseMax(float $vitesse):void
    {
        if ($vitesse >= 0) {
            $this->vitesseMax = $vitesse;
        } else {
            $this->vitesseMax = 0;
        }
    }

    public function getVitesseMax():float
    {
        return $this->vitesseMax;
    }
}
```

On peut voir dans cet exemple que l'utilisation d'un setter nous permet de contrôler la valeur vitesse. Ainsi il ne sera pas possible de stocker une vitesse négative.

On peut typer les arguments ainsi que le retour.



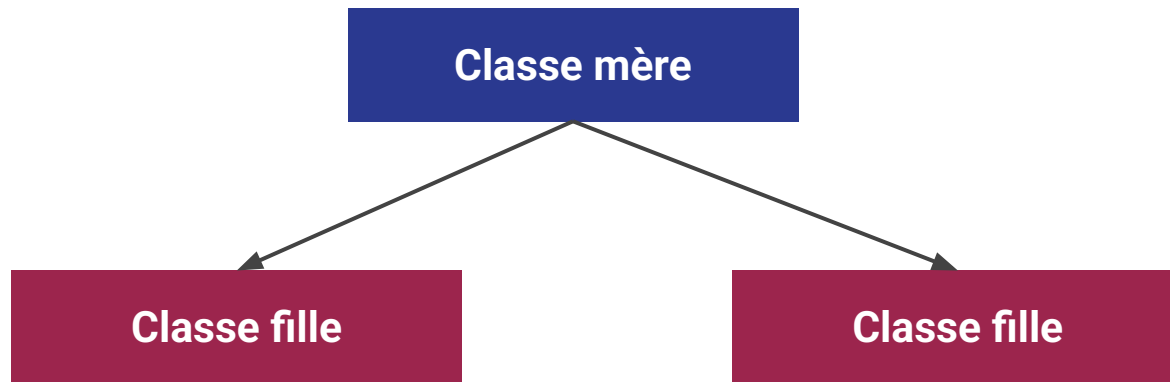
Exercice 12 suite

- Modifier la classe rectangle Rectangle.php
 - Passer toutes les propriétés en protected
 - Ajouter getters et setters (ne pas autoriser de longueur ou largeur négative)
- Dans index.php
 - Appeler les getters pour afficher toutes les informations

Héritage

Le concept d'héritage permet de créer un lien entre une classe mère et une classe fille. La classe fille héritera ainsi les propriétés et méthodes présente dans la classe mère.

La classe fille pourra ajouter des propriétés et méthodes.



Héritage

```
class Vehicule
{
    protected string $marque;
    protected float $vitesseMax;
}

class Voiture extends Vehicule {
    protected int $roues;
}

class Bateau extends Vehicule {
    protected int $nombreCabines;
}
```

Ici on peut voir la classe mère Vehicule qui contient des propriétés.

Avec le mot clé **extends**, on peut préciser que Voiture hérite de Vehicule. La classe Voiture héritera des propriétés marque et vitesseMax et ajoute une propriété roues.

Pour avoir accès à une propriété ou à une méthode appartenant à l'enfant, ceux-ci doivent être **protected** et non **private**.

Polymorphisme par héritage : surcharge

```
class Animal
{
    public function makeSound() {
        echo "L'animal fait un son";
    }
}
class Dog extends Animal
{
    public function makeSound() {
        echo "Le chien aboie";
    }
}
class Cat extends Animal
{
    public function makeSound() {
        echo "Le chat miaule";
    }
}
```

Lorsqu'une classe fille hérite des méthodes d'une classe mère, elle a la possibilité de redéfinir le comportement de cette méthode. On appelle cela la surcharge de méthode ou "overriding".



Abstraction


```
abstract class Animal
{
    public abstract function makeSound():void;
}
class Dog extends Animal
{
    public function makeSound():void {
        echo "Le chien aboie";
    }
}
class Cat extends Animal
{
    public function makeSound():void {
        echo "Le chat miaule";
    }
}
```

Dans l'exemple précédent, on se rend compte que la classe `Animal` n'a pas d'utilité d'être instancié car on ne veut pas créer un objet `Animal` mais plutôt directement le bon animal (chien ou chat).

De même la méthode `makeSound` n'a pas vraiment d'intérêt.

Dans ce cas, on fait une **classe abstraite** qui ne pourra pas être instanciée. Si on ajoute des **méthodes abstraites** elles seront **sans code**.

Les classes filles contraintes d'implémenter les méthodes de la classe mère.



Exercice 12 suite

- Continuer dans le dossier **exercice_formes**
- Créer un nouveau fichier `Forme.php`
 - Ajouter un constructeur avec couleur (et getter/setter)
 - Dans ce fichier faire une classe abstraite avec une méthode abstraite `calculerSurface()`
- Modifier la classe `Rectangle` (en supprimant `getCouleur` et `setCouleur`) et hériter de `Forme` puis vérifier si la page index s'ouvre sans erreur
- Créer un nouveau fichier `Cercle.php` qui hérite de `Forme`
- Dans ce fichier créer une classe du même nom avec :
 - propriété `rayon`
- Dans le fichier `index.php`
 - Importer `Cercle` et faire faire un nouvel objet
 - Constater l'erreur
- Dans `Cercle`, ajouter la méthode `calculerSurface()` qui retourne $\pi * \text{rayon}^2$ et appeler cette méthode dans `index.php`

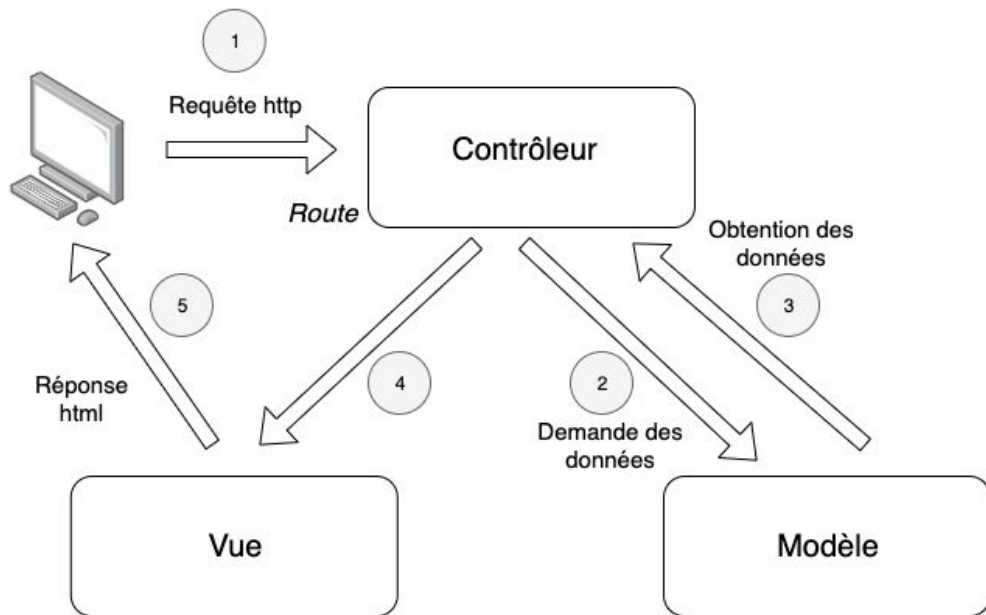
Modèle-Vue-Contrôleur

MVC : Définition

Le modèle MVC (Modèle-Vue-Contrôleur) est un modèle de conception pour le développement de logiciels qui permet de **séparer les responsabilités** des différentes parties d'une application en trois parties : **le modèle, la vue et le contrôleur**.



Modèle-Vue-Contrôleur (MVC)



MVC

- **Modèle** : Le modèle **représente les données** de l'application et les règles métier associées. Il contient les méthodes qui permettent de manipuler ces données, ainsi que les fonctions de validation, de transformation et de persistance.
- **Vue** : La vue représente l'**interface utilisateur de l'application**, c'est-à-dire la manière dont les données sont présentées à l'utilisateur final. Elle est responsable de l'affichage des données, de leur mise en forme et de leur organisation.
- **Contrôleur** : Le contrôleur agit comme l'**intermédiaire** entre le modèle et la vue. Il gère les interactions utilisateur (requêtes), récupère les données du modèle et les transmet à la vue.

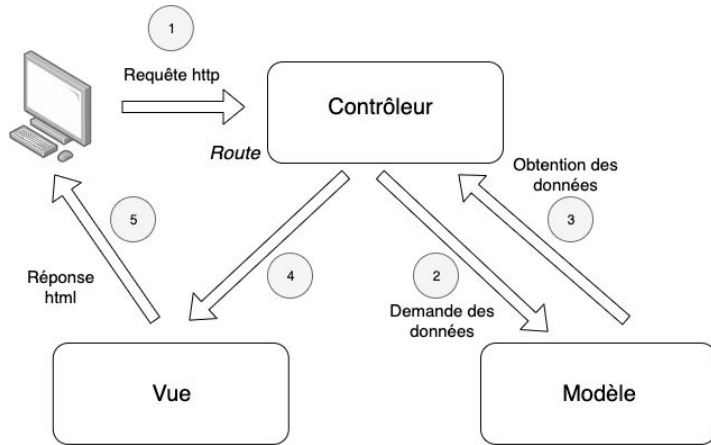


Importance de la séparation des responsabilités

- La séparation des responsabilités entre le modèle, la vue et le contrôleur est une pratique très courante dans le développement d'applications web et logiciel.
- Cette organisation permet de faciliter la conception, la maintenance et l'évolution des applications.
- Elle permet également d'améliorer la lisibilité et la compréhensibilité du code, en le découpant en composantes indépendantes et cohérentes.



MVC



Exemple pour une page article

- 1. Un utilisateur demande un article
- 2. Le contrôleur `ArticleController.php` récupère les informations de la requête (ex: `id_article`) et fait la demande au modèle `Article.php` pour récupérer les données
- 3. `Article.php` retourne les données à `ArticleController.php`
- 4. `ArticleController.php` retourne les données à la vue et charge la vue `article.php` qui contient le html
- 5. La vue complète est retournée à l'utilisateur