

I01 et I02

Langages de programmation

-

PHP

Table des matières

Présentation	5
Quelques détails techniques	5
Langage interprété VS langage compilé	5
Langage haut niveau.....	5
Références	6
Installation.....	7
Visual Studio Code.....	7
MAMP.....	7
Création du premier projet	9
Partie 1 - Syntaxe et sémantique	14
La base.....	14
Ecrire un commentaire	15
Debugger son code.....	15
Les variables	15
Noms de variables	15
Types de variables	16
Opérations mathématiques	17
Les comparateurs	18
Égalité	18
Inégalité.....	18
Ternaire	18
OR, AND, NOT.....	18
Le conditionnel	19
Switch	19
Les fonctions.....	19
Opérations sur les nombres	20
Opérations mathématiques de base via Math.....	21
Opération sur les chaînes de caractères (string).....	21
Documentation des fonctions	21
Les tableaux (array)	22
Fonctions sur les tableaux	23
Les boucles	24
While	25

Do...while	25
For.....	25
Foreach	25
La portée des variables.....	26
Global	26
Local.....	26
Static.....	27
Include	27
Les dates	27
Format	Erreur ! Signet non défini.
Exercices - partie 1	29
Partie 2 – Avancé.....	30
Les requêtes GET vs. POST	30
GET.....	30
POST	31
Formulaire	31
La session.....	31
Les exceptions	33
Les expressions régulières.....	34
Exercices - partie 2	34
Liste de courses	34
Jeu de carte	35
Partie 3 – Pour aller plus loin	36
La programmation orientée objet.....	36
Les classes.....	36

Présentation

Le PHP (PHP: Hypertext Preprocessor) est un langage de programmation dont la première version est sortie en 1994. Il est principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP. C'est un langage interprété (voir ci-dessous).

Un site Web, c'est globalement un serveur (back-end) et un client (front-end) qui communiquent. Quand un client (un utilisateur) demande au serveur d'ouvrir une page web, le serveur lui renvoie des informations comme par exemple la page HTML à afficher. Celle-ci est alors statique quand le contenu ne change pas. Le contenu est dynamique quand il varie en fonction de l'utilisateur, des données de la base de données, ... Le PHP permet de créer ce dynamisme en modifiant le code HTML renvoyé au front-end.

Il est le langage de programmation web côté serveur (back-end) le plus utilisé, représentant en 2016 82% des parts de marché (<https://w3techs.com/>), principalement grâce aux CMS tels que WordPress, Shopify, Joomla, Wix, ...)

Quelques détails techniques

Le PHP a connu plusieurs versions au cours du temps. Ce cours se concentrera sur la dernière version stable au moment de la rédaction de celui-ci (2020), la version 7.4. La version 8 est néanmoins prévue pour fin 2020.

Langage interprété VS langage compilé

Le code d'un langage interprété comme le JavaScript et le PHP est traduit lors de chaque exécution par un interpréteur. Les navigateurs internet sont des interpréteurs JavaScript. Il faut donc utiliser un interpréteur PHP pour exécuter le code PHP. Les langages compilés (C, BASIC, COBOL, ...) sont traduits une fois pour toute par le compilateur et peuvent ensuite être exécutés sans interprète. Le Java est situé entre les deux. Voir [Machine virtuelle Java](#)).

Etant donné qu'un langage compilé n'a pas besoin d'un interpréteur, il sera en général plus rapide.

Un langage interprété peut fonctionner sur tous les environnements pour autant qu'il existe un interpréteur compatible, ce qui n'est pas le cas d'un langage compilé. Par exemple, les .exe ne fonctionnent pas sur MacOS. Parfois pour un même OS, le type d'architecture du processeur ne permet pas le fonctionnement de l'exécutable (architecture 64-bit / 32 bits, ...). Il faut donc recompiler le code en fonction des types d'architecture souhaités. Cela nécessite généralement une adaptation du code.

Langage haut niveau

Le PHP est un langage haut niveau (tout comme le JavaScript). C'est à dire qu'il fonctionne au plus près de l'utilisateur et plus loin du fonctionnement de l'OS (système d'exploitation). Il utilise des mots

proches du langage naturel, des symboles mathématiques et fait abstraction du matériel utilisé. Un langage comme le C demandera plus de gestions des ressources (RAM, ...) alors que ce sera moins vital en PHP. Les langages encore plus bas niveau que le C, comme l'assembleur, sont au plus proche du langage de la machine et donc peu lisibles aisément pour des non-initiés.

Références

Tutoriel (français)

<https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql>

Tutoriel qui résume rapidement les différents éléments de PHP (anglais facile)

<https://www.w3schools.com/php/default.asp>

Un premier exemple et une excellente page pour tester des petits scripts :

https://www.w3schools.com/php/phptryit.asp?filename=tryphp_intro

La documentation officiel (français) :

<https://www.php.net/manual/fr/tutorial.php>

Installation

Avant toute chose, il faut installer l'environnement de travail.

Du code PHP peut s'écrire avec n'importe quel éditeur de texte. Néanmoins, cela s'avère très vite insuffisant, surtout avec de plus grands projets. Il est conseillé d'utiliser un éditeur de code tel que PHPStorm ou Visual Studio Code. Le premier étant payant, le deuxième lui sera préféré.

Il faut également installer un serveur pour interpréter le PHP et simuler le comportement des pages web. Il existe différentes solutions d'environnement de développement gérant aussi la base de données comme WAMP (Windows) ou MAMP (Macintosh). MAMP sera préféré car il est compatible avec Mac OS et Windows (pour Linux : <https://doc.ubuntu-fr.org/lamp>)

Visual Studio Code

Téléchargez l'éditeur via : <https://code.visualstudio.com/>

Installez-le en laissant toutes les options par défaut.

MAMP

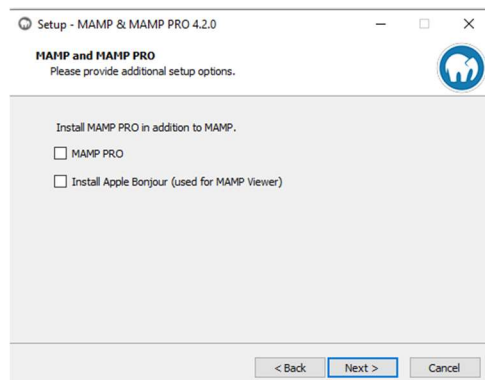
MAMP signifie **M**acintosh **A**pache **M**ySQL **P**HP, WAMP pour Windows, et LAMP pour Linux.

- Apache est un serveur HTTP qui permet également d'interpréter le PHP.
- MySQL est le SGBD (gestionnaire de base de données) qui permet de gérer facilement la base de données utilisée avec le code PHP.

Téléchargez MAMP via :

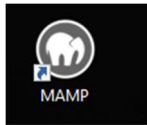
<https://www.mamp.info/en/downloads/>

Décochez les options ci-dessous :



Laissez les autres options par défaut.

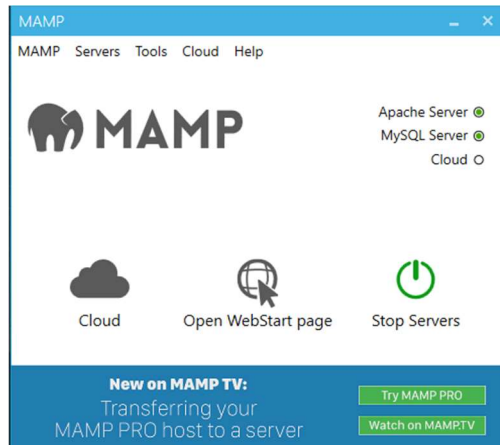
Démarrez MAMP en cliquant sur l'icône.



Cliquez ensuite sur Start Servers.




Si tout va bien, Apache Server et MySQL server s'affichent en vert :




Appuyez sur Open WebStart page, ce qui doit ouvrir votre navigateur préféré et afficher :

[Start](#) [My Website](#) [phpInfo](#) [Tools](#) [FAQ](#) [MAMP Website](#) [Buy MAMP PRO](#)



Welcome!

MAMP has been installed successfully.



PHP

[phpinfo](#) shows the current configuration of PHP.

MySQL

MySQL can be administered with [phpMyAdmin](#).

To connect to the MySQL server from your own scripts use the following connection parameters:

Host	localhost
Port	3306
User	root
Password	root

Examples

[PHP <= 5.5.x](#) [PHP >= 5.6.x](#) [Python](#) [Perl](#)

```
$user = 'root';
$password = 'root';
$db = 'Inventory';
```

MAMP Version

4.2.0 → Your version is up-to-date.

News

[MAMP & MAMP PRO 4.1.1 out now](#)

February, 2017 – **Version 4.1.1 is out now - comes with German localization**

- Added german localisation
- Initial TouchBar support on MacBook Pro 2016
- Automatically deleting a database when deleting a host fixed
- Some functions can now only be used if the active servers are using the current configuration
- Solved PHP 7.1 configuration quirks

New components:

- MySQL 5.6.35
- PHP 5.6.29, 5.6.30, 7.0.14, 7.0.15 & 7.1.1
- Python 2.7.13
- libpng 1.6.27
- phpMyAdmin 4.6.5.2
- curl 7.52.1

[Happy New Year! MAMP & MAMP PRO 4.1 has just been released](#)

Différentes informations sont visibles sur cette page, dont les informations de connexions à la base de données MySQL.

MySQL

MySQL can be administered with [phpMyAdmin](#).

To connect to the MySQL server from your own scripts use the following connection parameters:

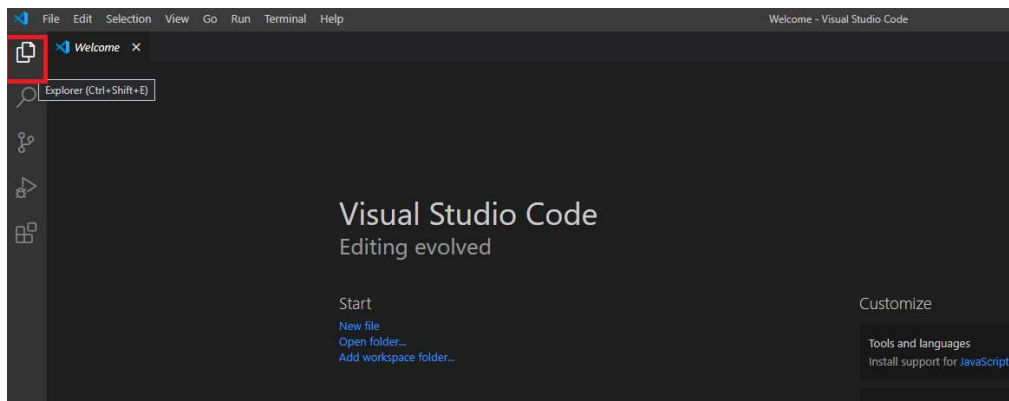
Host	localhost
Port	3306
User	root
Password	root

Cliquez sur le lien bleu [phpMyAdmin](#).

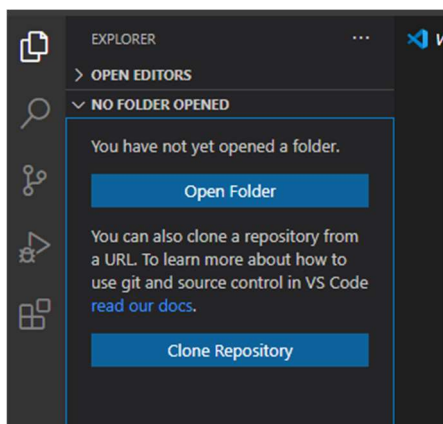
Une interface permettant de configurer cette base de données (créer un schéma, ajouté des tables, ...) s'ouvre.

Création du premier projet

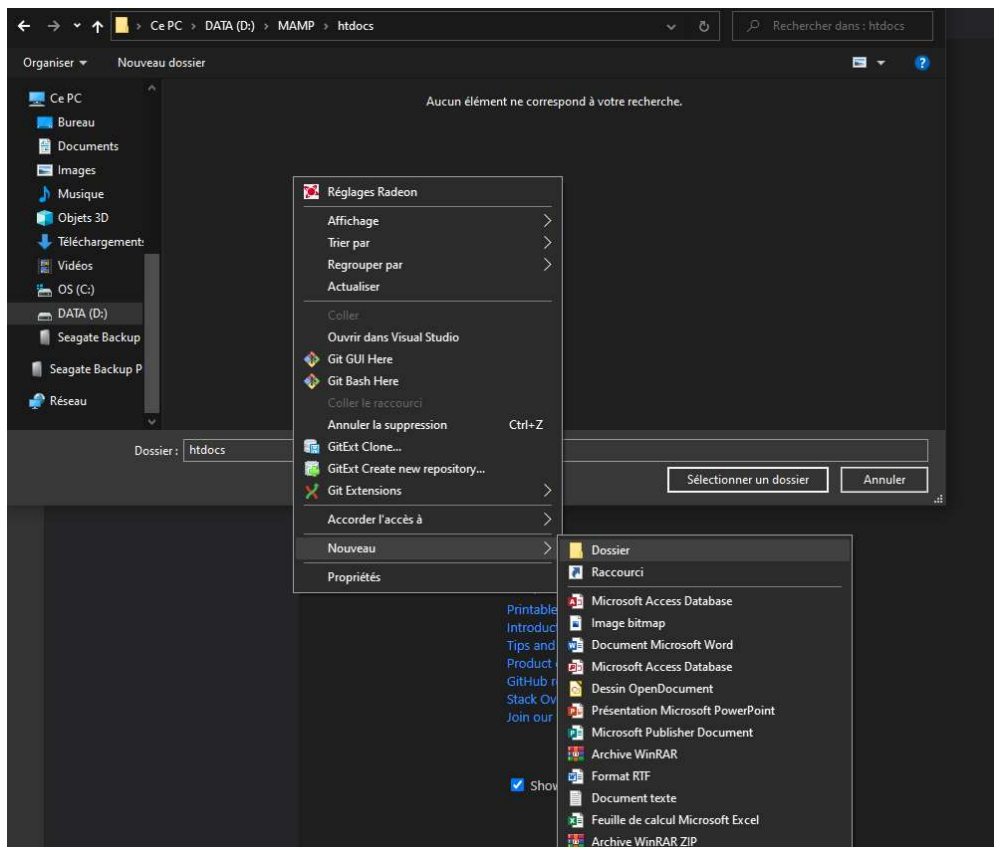
Ouvrez Visual Studio Code. Cliquez sur explorer en haut à gauche.



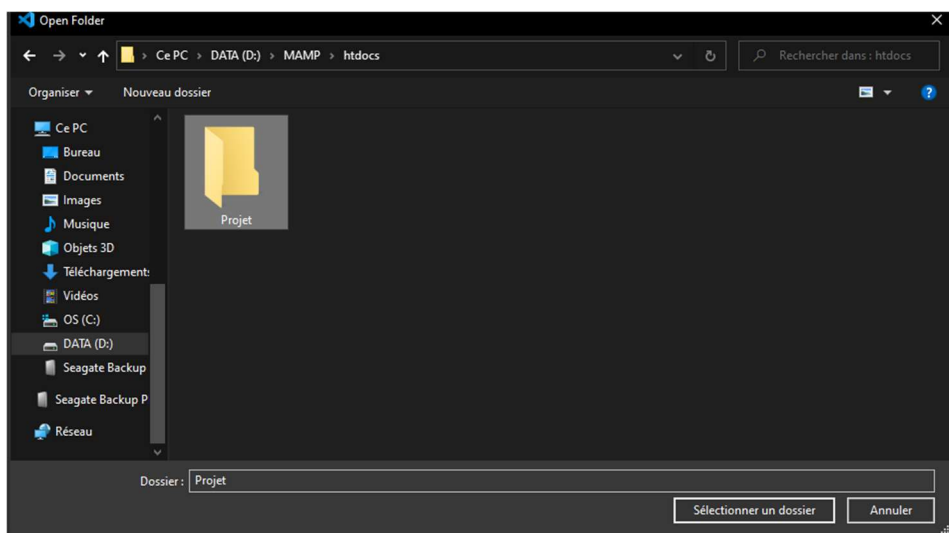
Cliquez sur Open Folder



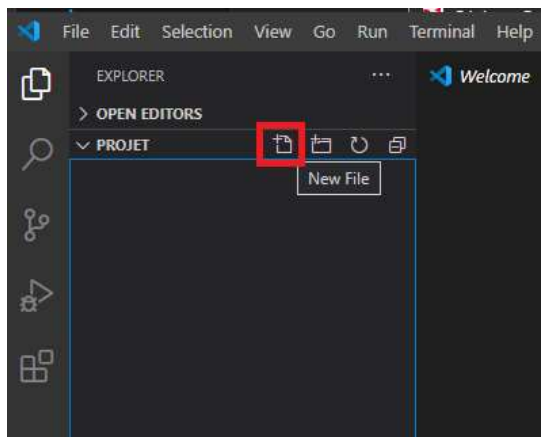
Allez dans le dossier d'installation MAMP, puis dans le dossier htdocs (normalement C:\MAMP\htdocs)



Faites clique-droit dans l'explorateur de fichiers et créez un nouveau dossier, nommez-le « Projet » par exemple. Sélectionnez-le et appuyez sur Sélectionner un dossier.



Cliquez sur New File et le nommez index.php

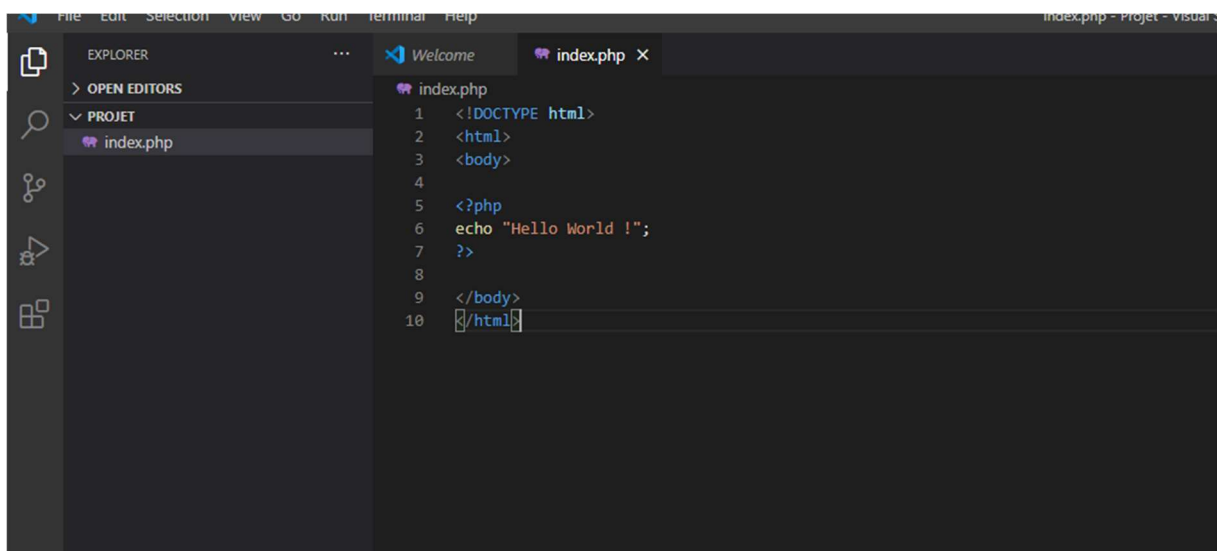


Copiez le code suivant dans le fichier :

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "Hello World !";
?>

</body>
</html>
```



Sauvegardez le fichier (ctrl+s).

Enfin, dans votre navigateur préféré, tapez l'url : <http://localhost/projet/>. Vous devriez obtenir le résultat suivant :

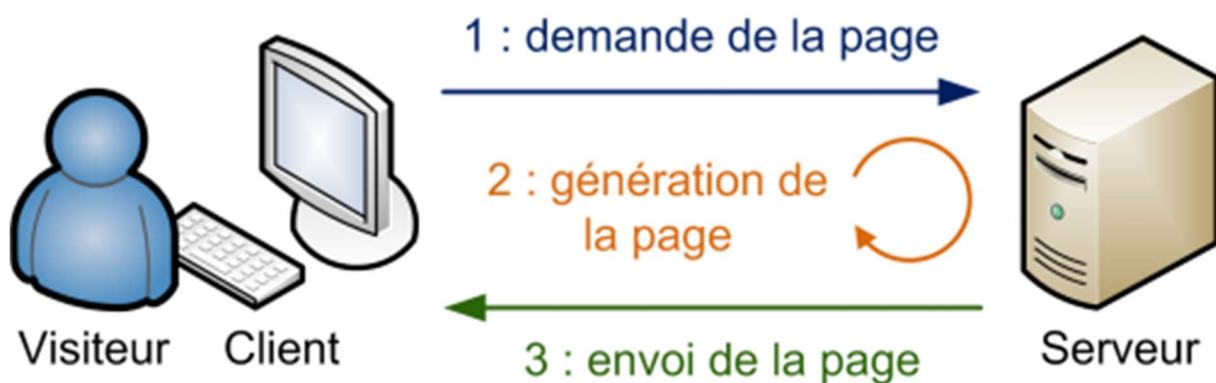


Vous pourrez modifier le code

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

avec le code PHP de votre choix et recharger la page <http://localhost/projet/> afin de tester votre code.

La communication entre le client et serveur et génération du PHP peuvent être schématisés comme ceci :



Partie 1 - Syntaxe et sémantique

La syntaxe d'un langage informatique est la manière dont les différents éléments du langage peuvent être assemblés pour former un programme fonctionnel. Par exemple la manière dont une boucle (while, for, ...) peut être combinée avec une instruction conditionnelle (if).

La sémantique est la signification des phrases utilisées dans un langage. Par exemple, « $a > b ? a : b$; » donne en français « si a est plus grand que b, renvoi a, sinon renvoi b ».

Nous allons donc voir le vocabulaire utilisé pour écrire du code PHP, la signification et la manière d'agencer les éléments. Notre objectif sera de créer des pages Web dynamiques.

La base

Un script PHP peut être placé n'importe où dans un document. Il doit être écrit entre les balises `<?php` `?>`

Ouvrir la balise « `<?php` » avant tout code PHP est obligatoire pour qu'il soit compris. Fermer la balise n'est pas obligatoire, mais recommandé. Il est utile de la fermer lorsque du code PHP est inséré dans du code HTML.

```
<!DOCTYPE html>
<html>
<body>

<h1>Titre de la page</h1>

<?php
echo "Bonjour !";
?>

</body>
</html>
```

Le code PHP ne doit pas nécessairement être inséré dans du code HTML. Attention, le code PHP ne sera pas interprété par votre navigateur. Si vous essayez d'ouvrir test.html ou test.php (sur le repository GIT du cours), le navigateur ignorera le code PHP. Pour cela, il faut passer par un interpréteur PHP (voir installation avec MAMP par exemple).

Le code PHP n'est pas sensible à la casse. C'est-à-dire qu'il peut être écrit en minuscule ou majuscule, cela ne change rien à son interprétation.

```
ECHO "Bonjour !";
echo "Bonjour !";
EcHo "Bonjour !";
```

Attention, les noms des variables sont quant à eux sensibles à la casse (sujet abordé un peu plus loin) !

Ecrire un commentaire

Il est possible d'écrire un commentaire, c'est à dire du code qui sera ignoré par le navigateur.

```
// ceci est un commentaire sur une ligne
# ceci est aussi un commentaire sur une ligne
/*
    Ceci est un bloc commentaire, on peut écrire sur plusieurs lignes.
    Les commentaires sont utiles pour attirer l'attention de la personne
    lisant le code (soi-même ou d'autres personnes) sur un élément qui
    pourrait être difficile à comprendre, ou simplement pour expliquer le
    fonctionnement d'un bout de code.
*/
```

Cela permet de documenter son code, ce qui est souhaitable, par exemple en décrivant une fonction ou une méthode donnée, le rôle des paramètres d'entrée et ce que la fonction va renvoyer.

Documenter et commenter son code sont deux choses primordiales pour s'assurer que son code puisse être lisible et compréhensible par tous et même soi-même. Cela permet de travailler plus facilement en équipe et de déboguer plus efficacement du code. [Les fonctions, qui sont commentées et documentées, sont abordées plus tard dans le cours. Plus d'explication sur le document dans le chapitre sur les fonctions.](#)

Debugger son code

Pour voir pourquoi votre code ne fonctionne pas, il faut se rendre dans le dossier C:\MAMP\logs. Les erreurs liées à votre code PHP sont indiquées dans le fichier php_error.log.

Pour afficher directement les erreurs sur la page HTML où vous êtes en train de travailler : <https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql/4238821-configurez-php-pour-visualiser-les-erreurs>

Les variables

Une variable permet de stocker une donnée ou un ensemble de données pouvant être traités.

Noms de variables

En PHP, on écrit une variable comme ci-dessous :

```
$x = 5; // Variable de type nombre entier
$txt = "Bonjour!"; // Variable de type chaîne de caractère
```

Le nom d'une variable PHP :

- commence toujours par un signe \$,
- peut pas commencer par un chiffre,
- ne peut pas contenir de caractères spéciaux (ex : é, @, ù, ...),
- doit absolument commencer par une lettre ou _.

Il est sensible à la casse, donc \$txt != \$TxT. Il est préférable que le nom commence par une minuscule et que chaque autre mot qui compose le nom de la variable commence par une majuscule. Il est également préférable que le nom de la variable soit explicite quant à son utilité.

Alors que dans la plupart des langages de programmation un mot clé est utilisé pour déclarer une variable, ce n'est pas le cas en PHP (par exemple en JS : `var myVariable`).

Types de variables

Il existe différents types de variables. Voici une partie des types de base :

- array => tableau (voir le chapitre sur les tableaux),
- bool => un booléen, true ou false, 1 ou 0,
- float => un nombre à virgule,
- int => un nombre entier,
- string => une chaîne de caractère.

Ces définitions de types sont principalement utilisées dans les fonctions (voir chapitre plus loin). Il est donc préférable d'utiliser ces types de données pour exiger un type de variable précis.

Le PHP est faiblement typé. Cela signifie qu'il n'est pas nécessaire de définir le type d'une variable (par exemple : une variable qui gère des nombres entiers, des variables de chaînes de caractères, ...). Le PHP gère cela automatiquement en fonction de la valeur (plus précisément du type de données de la valeur) qui a été assignée à la variable.

Néanmoins, avec PHP 7, des types de variables ont été ajoutés. Cela s'appelle le mode strict, ce qui va jeter une exception « Fatal Error » si le type de données n'est pas respecté (un chapitre du cours est dédié aux exceptions).

Il faut néanmoins savoir que même s'il ne faut pas définir de type de variable en PHP (ce n'est pas le cas dans tous les langages, par exemple en Java, C, ...), les variables ont bien un type en fonction de leurs valeurs (même chose que ci-dessus). Il faut donc y faire attention quand on les manipule. Le PHP possède un mécanisme de conversion implicite, qui permet dans certains cas de transformer une chaîne de caractères en nombre par exemple.

Variables textuelles

Une valeur ou une variable textuelle est appelée un string.

```
$texte = "text"; // on assigne une valeur textuelle à la variable
texte. " " et ' ' sont équivalents.
$texte = 'text'; // on réassigne une nouvelle valeur à cette variable.

$texte = 'bonjour' . ' ' . "le monde"; // permet de fusionner
plusieurs string, on appelle cela la concaténation.
```

Attention, dans la plupart des autres langages la concaténation des string se fait avec le signe +. Mais ce signe est réservé aux additions en PHP. Donc si vous essayez de faire `"bonjour" + "le monde"` le résultat sera 0 car PHP n'arrive pas à effectuer l'addition.

Opérations mathématiques

```
$nomDeLaVariable = 3; // on assigne la valeur entière 3 à la variable.
$nomDeLaVariable += 3; //on ajoute 3 à la valeur de la variable (3+3,
donc 6).
$nomDeLaVariable = $nomDeLaVariable + 3; // écriture plus longue mais
équivalente.

$n = 3 - 3 + (2 * (5 / 2)); // un exemple d'opération mathématique
écrite en PHP.

$n++; // on incrémente de 1 la valeur de la variable x, ce qui est
équivalent à x += 1 ou x = x + 1.
$n--; // on décrémente de 1 la valeur de la variable x, ce qui est
équivalent à x -= 1 ou x = x - 1.

2**2; // Exponentiel, 2 exposant 2.
11%2; // Modulo, c'est-à-dire le reste du résultat de la division.
Dans ce cas-ci 1. Car 11 = 5 x 2 + 1
```

Exercices

Quels seront les résultats des opérations suivantes :

Rappel : une chaîne de caractères s'écrit entre `" "` ou `' '`.

```
<?php

echo 5 + 5;

echo 5 + "5";

echo 5 * "5";

echo 5 - "5";
```

```

echo 5 . "5";

echo 5 + "-5";

echo 5 + "a5";

echo 5 - "a5";

echo "x" - "a5";

echo "10" - "5";

$a = 2;
$b = 3;

echo ($a ** $b) < 16

?>

```

Les comparateurs

Soit

```

$a = 1;
$b = 2;
$aText = '1';

```

Égalité

```

$a == $b; // retourne false
$b == $aText; // retourne false
$a == $aText; // retourne true

```

Inégalité

```

$a != $b; // retourne true
$b != $aText; // retourne true
$a != $aText; // retourne false

$a < $b; // retourne true
$a <= $b; // retourne true
$a <= $aText; // retourne true

```

Ternaire

```

$a > $b ? "si true alors on affiche ceci" : "si false on affiche
cela"; // retourne si false...

```

OR, AND, NOT

```
$a < $b && $a == $aText; // retourne true. && = ET (AND), ce qui veut
dire en français : si a est plus petit que b ET que a égal aText. Les
deux conditions doivent être vraies.
$a < $b || $a == $aText; // retourne true. || = OU (OR), ce qui veut
dire en français : si a est plus petit que b OU que a égal aText.
Seule une seule condition vraie suffit.
!($a < $b); // a < b = true et le ! (NOT) devant signifie l'inverse
du résultat, donc retourne false.
```

Le conditionnel

```
if($a < $b) {
    // la condition est vraie.
    echo "Vrai";
} else {
    // la condition est fausse.
    echo "Faux";
}
```

Switch

Un switch permet de soulager l'écriture des conditionnelles IF quand il y a beaucoup de cas possibles.

Dans le code suivant, la variable \$jourDeLaSemaine est le nom d'un jour de la semaine. Pour convertir celui-ci en un chiffre (exemple lundi = 1, mardi = 2, ...) :

```
switch ($jourDeLaSemaine) {
    case 'lundi' :
        echo 1;
        break;
    case 'mardi' :
        echo 2;
        break;
    case 'mercredi' :
        echo 3;
        break;
    case 'jeudi' :
        echo 4;
        break;
    case 'vendredi' :
        echo 5;
        break;
    case 'samedi' :
        echo 6;
        break;
    case 'dimanche' :
        echo 7;
        break;
}
```

Les fonctions

Une fonction est un bloc de code exécutant une tâche bien précise. Elle peut avoir des valeurs d'entrée (c'est généralement le cas en informatique) , appelées arguments ou paramètres de la fonction, et

peut retourner un résultat, la valeur de retour. Les fonctions sont particulièrement utiles pour découper le code en plusieurs sous-tâches afin de le rendre plus lisible et de pouvoir le réutiliser facilement afin d'éviter la duplication de code (du code qui fait plusieurs fois la même chose).

Par exemple, une fonction de conversion qui convertit les degrés fahrenheit en degrés Celsius :

```
function toCelsius($fahrenheit) {  
    return (5 / 9) * ($fahrenheit - 32);  
}  
  
echo toCelsius(10); // affiche -12.22222222222223
```

Il est en effet plus facile d'écrire le calcul une fois pour toute que de le répéter plusieurs fois. Même s'il n'est fait qu'une fois, il est toujours bon d'« isoler » le calcul dans une fonction. Cela permet de le réutiliser quand on le souhaite, mais aussi de le retrouver plus facilement dans le code et de rendre l'utilité du calcul plus intelligible (grâce au nom de la fonction par exemple).

A chaque fois que vous avez l'impression que vous allez utiliser plusieurs fois un même bout de code, que vous avez déjà écrit un bout de code quelque part ou encore qu'un bout de code commence à devenir trop volumineux, il faut découper votre code en plus petites fonction. Seul l'expérience pourra vous donner le bon feeling à ce niveau. Mais la règle est que l'informaticien est paresseux, ce qui signifie qu'il n'aime pas faire quelque chose qui a déjà été fait et qu'il pourrait réutiliser (que ce soit par lui-même ou par quelqu'un d'autre).

Il existe un tas de fonctions déjà écrites en PHP : des opérations sur les nombres, les chaînes de caractères, les tableaux, les dates, ...

Exemple :

```
var_dump($myVar)
```

Cette fonction retourne la valeur de la variable, ce qui est utile pour déboguer. Voir différence entre `echo $myVar` et `var_dump($myVar)` sur un tableau pour mieux comprendre son utilité.

Opérations sur les nombres

```
is_int($x); // Retourne TRUE si la variable en paramètre est un nombre entier  
is_float($x); // Retourne TRUE si la variable en paramètre est un nombre décimal  
is_finite($x); // Retourne TRUE si la variable en paramètre est un nombre fini  
is_infinite($x); // Retourne TRUE si la variable en paramètre n'est pas un nombre fini  
is_nan($x); // Retourne TRUE si la variable en paramètre n'est pas un nombre  
// Utile pour les opérations mathématique impossible, exemple :  
$x = acos(8); (https://www.w3schools.com/php/func\_math\_acos.asp)  
is_numeric($x); // Retourne TRUE si la variable en paramètre est un nombre (ou un string représentant un nombre correcte).
```

Opérations mathématiques de base via Math

`min()` et `max()` retournent le plus petit et le plus grand nombre d'un tableau donné (les tableaux sont abordés au point suivant).

```
$arr = [0, 150, 30, 20, -8, -200];
echo(min(0, 150, 30, 20, -8, -200)); // retourne -200
echo(max($arr)); // retourne 150
abs(-5); // retourne la valeur absolue du nombre en paramètre, ici 5
sqrt(4); // retourne la racine carrée du nombre en paramètre, ici 2
round(4.49); // retourne le nombre en paramètre arrondi, ici 4
round(4.59); // retourne 5
rand(); // retourne un nombre entier aléatoire
rand(0, 100); // retourne un nombre entier entre [0 et 100] (inclus)
```

Liste des fonctions mathématiques : <https://www.php.net/manual/fr/book.math.php>

En savoir plus sur les nombres pseudo-aléatoires :

https://fr.wikipedia.org/wiki/Générateur_de_nombres_pseudo-aléatoires

Opération sur les chaînes de caractères (string)

```
strlen("Hello world!"); // retourne le nombre de caractère dans le
string en paramètre. Ici : 12
str_word_count("Hello world!"); // retourne le nombre de mots dans le
string en paramètre, ici 2
strpos("Hello world!", "world"); // retourne l'index du mot en 2ème
paramètre, contenu dans le premier string en paramètre . Ici 6.
str_replace("world", " I01 et I02", "Hello world!"); // remplace le
premier string, par le 2ème dans le 3ème string. Résultat : Hello I01
et I02!
trim("   world   "); // retourne le string en supprimant les
espaces inutiles en début et fin de string, ici : "world"
substr_count("Hello world world!", "world"); // retourne le nombre
d'occurrences du deuxième paramètre dans le premier. Ici : 2
substr("Hello world!", 6, 5); // retourne le string en premier
paramètre à partir de l'index 6 pour une longueur de 5 caractères.
Résultat : world.
explode ("world", "Hello world world!"); // retourne un tableau
contenant les éléments du string en 2ème paramètre, séparé par le
premier string. Résultat : ["Hello ", " ", "!"]
explode (" ", "Hello world world!"); // Résultat : ["Hello", "world",
"world!"]
```

Liste des fonctions sur les chaînes de caractères : <https://www.php.net/manual/fr/ref.strings.php>

Documentation des fonctions

Ecrire de la documentation, c'est donner la description d'une fonction ou une méthode donnée, de ses paramètres d'entrée et sa valeur de retour. La documentation permet également de décrire beaucoup plus, par exemple les préconditions et postconditions.

Les préconditions sont des conditions qui s'appliquent aux paramètres de la fonction et qui doivent être vraies avant l'exécution de la fonction (par exemple une chaîne de caractères non vide).

Les postconditions sont des conditions qui s'appliquent à la valeur de retour d'une fonction et qui doivent être vraies après exécution de celle-ci (la valeur de retour doit être un entier positif).

Elles permettent de s'assurer que la fonction fait bien ce à quoi on s'attend. Cela est donc très utile pour déboguer son code. Nous ne les utiliserons néanmoins pas dans ce cours.

Voici un exemple de documentation pour la fonction de conversion fahrenheit / Celsius :

```
/**
 * Convertit une température reçue en degrés fahrenheit en degré
 * Celsius.
 *
 * @param $fahrenheit la température en degrés fahrenheit
 * @return float représentant la température en degrés Celsius
 */
function toCelsius($fahrenheit) {
    return (5 / 9) * ($fahrenheit - 32);
}
```

Les tableaux (array)

Les tableaux permettent de stocker plusieurs données en même temps, typiquement une liste d'objets (une liste de joueurs, d'items,...)

Exemple de déclarations d'un tableau avec des valeurs string (équivalent) :

```
$names = ["Martin", "Jérôme", "Henri"];
$names = array("Martin", "Jérôme", "Henri");
```

Déclaration d'un tableau vide :

```
$names = array();
```

Accéder à un élément du tableau :

```
$names[0]; // Retourne Martin
```

Connaitre le nombre d'éléments dans le tableau :

```
count($names); // Retourne 3
```

Si on essaye d'accéder à un élément qui n'est pas dans le tableau :

```
$names[3]; // Retourne null
```

Il est donc toujours préférable de vérifier la longueur d'un tableau avant d'accéder à un élément :

```
if(count($names) > 3){
    $names[3];
} else {
    /*
        La taille du tableau est = 3, car il y a 3 éléments. Or on essaye
        d'accéder au 4ème élément, ce qui ne peut pas fonctionner.
        En effet, le premier élément du tableau est accessible via $names[0].
        Il faut donc faire -1 pour accéder à l'élément souhaité.
        Ex : le 4ème élément : 4-1 = 3
    */
}
```

De manière classique, les tableaux sont indexés avec un nombre entier allant de 0 à la longueur du tableau – 1. En PHP (attention ce n'est pas souvent le cas dans les autres langages de programmation) les tableaux peuvent également être indexés avec un string.

Exemple :

```
$array["Martin"] = "test" ;
```

Cela peut être utile dans certains cas précis que nous verrons plus tard.

Fonctions sur les tableaux

```
count($array); // retourne le nombre d'éléments dans le tableau en
paramètre

$array = ['a', 'b', 'c', 'd', 'e'];
array_chunk($array, 2); // retourne un tableau avec le nombre
d'éléments en paramètre, ici ['a', 'b']

array_combine($array, array('f', 'g', 'h')); // retourne les 2
tableaux en paramètres en un seul tableau. Résultat : ['a', 'b', 'c',
'd', 'e', 'f', 'g', 'h']
```

Info : plus ou moins équivalent à array_merge (<https://www.php.net/manual/fr/function.array-merge.php>)

```
array_fill(0, 3, 'a'); // retourne un tableau rempli à partir de
l'index 0 avec 3 éléments avec la valeur donnée en 3ème paramètre. Ici
: ['a', 'a', 'a']

$arr2 = ['a', 'a', 'c', 'a', 'e'];
array_filter($arr2, function($v, $k) {
    return $k == 'a';
}, ARRAY_FILTER_USE_BOTH); // retourne un tableau correspondant à la
condition présente dans la fonction. Ici ['a', 'a', 'a']. $v = la
valeur de l'élément du tableau, $k = la clé (ou index)
```

<https://www.php.net/manual/fr/function.array-filter.php>

```

$arr3["Martin"] = "test" ;
array_key_exists('Martin', $arr3); // retourne TRUE si la clé est
contenue dans le tableau en paramètre

function cube($n)
{
    return ($n * $n * $n);
}
$a = [1, 2, 3, 4, 5];
array_map('cube', $a); // Retourne un nouveau tableau où on a effectué
une opération sur chaque élément du tableau
// Résultat : array(1, 8, 27, 64, 125);

array_pop($a); // retire le dernier élément du tableau. Retire donc
l'élément 5 à $a -> array(1, 2, 3, 4)

array_push($a, 6); // ajoute un (ou plusieurs) élément(s) à la fin du
tableau. // Ajoute 6 à $a -> array(1, 2, 3, 4, 6)

array_shift($a); // retire le premier élément du tableau. Retire donc
1 à $a -> array(2, 3, 4, 6)

array_unshift($a, 0); // ajoute un (ou plusieurs) élément(s) au début
du tableau. // Ajoute 0 à $a -> array(0, 2, 3, 4, 6);

$a = array(1, 2, 3, 4, 5);
array_rand($a); // retourne au hasard une clé présente dans le tableau
en paramètre. Ici une valeur entre [0, 4].

array_search(2, $a); // retourne l'index où se trouve l'élément en
paramètre du tableau. Ici : 1. Retourne false si rien n'est trouvé.

sort($a); // trie le tableau en paramètre dans l'ordre croissant :
array(1, 2, 3, 4, 5)

rsort($a); // trie le tableau en paramètre dans l'ordre décroissant :
array(5, 4, 3, 2, 1)

```

Plus de fonctions sur les tableaux : <https://www.php.net/manual/fr/ref.array.php>

Les boucles

Une boucle permet d'exécuter le même code plusieurs fois, tant qu'une condition est vraie.

Une boucle doit être constituée de trois étapes pour fonctionner :

1. Initialisation
2. Condition de sortie
3. Incrémentation

While

```
$i = 0; // initialisation
while ( $i < 10 ) // condition de sortie, doit se lire tant que $i
est plus petit que 10
{
    echo $i ; // affiche 0, 1, 2, .. jusqu'à 9
    $i++; // incrémentation, sinon la boucle est infinie
}
```

Do...while

```
$i = 0;
do
{
    echo $i++; // affiche 0, 1, 2, .. jusqu'à 9
} while ( $i < 10 ) ;
```

La condition de sortie est à la fin de l'instruction. On passe donc au moins une fois dans la boucle, même si à l'initiation $\$i > 10$.

For

Une autre façon d'écrire une boucle plus condensée :

```
for($j = 0; $j < 10; $j++) // initialisation, condition de sortie et
incrémentation dans la même ligne
{
    echo $j++; // affiche 0, 1, 2, .. jusqu'à 9
}
```

Les boucles sont particulièrement utiles pour traiter le contenu d'un tableau :

```
$fruits = ["Banane", "Orange", "Pomme", "Mangue"];
for($iFruit = 0; $iFruit < count($fruits); $iFruit++) {
    echo $fruits[$iFruit]; // affiche chaque élément du tableau
}
```

Foreach

Une autre façon de traiter un tableau plus condensé :

```
$fruits = ["Banane", "Orange", "Pomme", "Mangue"];
foreach($fruits as $fruit) {
    echo $fruit ; // affiche chaque élément du tableau dans la console
}
```

La portée des variables

La portée d'une variable (ou scope d'une variable) est la partie du script où la variable peut être référencée/utilisée.

Les 3 types de scopes en PHP sont :

- Global
- Local
- Static

Global

```
$x = 5; // global
function myTest() {
    // l'utilisation de x à l'intérieur de cette fonction générera une
    erreur
    echo "<p>Variable x dans la fonction : $x</p>";
}
myTest();
echo "<p>Variable x en dehors de la fonction: $x</p>";
```

Pour utiliser \$x dans la fonction test :

```
$x = 5; // global
function myTest() {
    global $x ;
    echo "<p>Variable x dans la fonction : $x</p>";
}
myTest();
echo "<p>Variable x en dehors de la fonction: $x</p>";
```

Local

```
function myTest() {
    $x = 5; // local
    echo "<p>Variable x dans la fonction: $x</p>";
}
myTest();
// utiliser x en dehors de la fonction générera une erreur
echo "<p>Variable x en dehors de la fonction: $x</p>";
```

Static

Normalement, lorsqu'une fonction est terminée/exécutée, toutes ses variables sont supprimées. Pour empêcher ce comportement, il faut utiliser le mot clé static :

```
function myTest() {  
    static $x = 0;  
    echo $x;  
    $x++;  
}  
  
myTest(); // affiche 0  
echo "<br>";  
myTest(); // affiche 1  
echo "<br>";  
myTest(); // affiche 2
```

Include

L'instruction include permet d'ajouter le code provenant d'un fichier PHP au contenu actuel.

Prenons l'exemple de test.php où il faut ajouter le contenu de vars.php :

```
===== vars.php  
<?php  
  
$couleur = 'verte';  
$fruit = 'pomme';  
  
?>
```

```
===== test.php  
<?php  
  
echo "Une $fruit $couleur"; // Une  
  
include 'vars.php';  
  
echo "Une $fruit $couleur"; // Une pomme verte  
  
?>
```

Les dates

En PHP, la fonction date(format, timestamp) permet de gérer les dates.

- format : requis, il spécifie le format sous lequel la date doit être affichée.
- timestamp : optionnel, c'est un nombre entier représentant le nombre de secondes depuis le 1er janvier 1970 00:00:00. Par défaut, le timestamp est l'heure actuelle.

Format :

- d – jour du mois (01 to 31)
- m - mois (01 to 12)
- Y - année (4 chiffres)
- l (lowercase 'l') – Nom du jour du mois
- H - 24-heure (00 to 23)
- h - 12-heure (01 to 12)
- i - Minutes (00 to 59)
- s - Secondes (00 to 59)
- a - (am ou pm)

```
echo date("Y/m/d") // affiche 2020/11/03
echo date("Y.m.d") // affiche 2020.11.03
echo date("Y-m-d") // affiche 2020-11-03
echo date("l") // affiche Tuesday
echo date("Y/m/d h:i:sa ") // affiche 2020/11/03 07:33:17pm
```

Voir l'heure UNIX : https://fr.wikipedia.org/wiki/Heure_Unix

Plus d'infos sur les dates : <https://www.php.net/manual/fr/function.date.php>

Exercices - partie 1

<https://www.codingame.com/playgrounds/32339/exercices-de-php-pour-debutant>

Partie 2 – Avancé

Les requêtes GET vs. POST

Pour rappel, la communication entre le client et le serveur se fait comme ceci :



En informatique, quand plusieurs acteurs communiquent sur le réseau, ils utilisent des protocoles pour se comprendre. Dans le cas du Web, c'est le protocole HTTP qui est utilisé. Celui-ci permet de communiquer via plusieurs méthodes. Nous allons nous concentrer sur les plus utilisées, GET et POST.

GET

Il permet à l'utilisateur d'envoyer au serveur une demande pour une page spécifique avec certains paramètres.

Par exemple dans ce lien : <https://www.youtube.com/watch?v=US9JCsnAVTU>

- youtube.com est le nom de domaine à qui adresser la requête (vers quel serveur envoyer la requête).
- /watch est la page demandée
- ? après le point d'interrogation, ce sont les données que l'on veut envoyer au serveur. Ici on dit que le paramètre v = US9JCsnAVTU, qui est le code de la vidéo que l'on souhaite regarder.

GET est donc la manière dont on communique classiquement sur le Web. Quand vous entrez une URL dans votre navigateur, vous effectuez une requête HTTP avec la méthode GET.

La méthode GET est donc visible par tous (tous les paramètres, les valeurs, ...). Ce qui est bien pour partager des URLs par exemple. Pour envoyer des données sensibles aux serveurs, un mot de passe par exemple, il faudra pour cela utiliser POST.

La taille des données que l'on peut envoyer avec GET est limitée à 2048 caractères. Pour envoyer de plus grandes données, ou des données non textuelles comme une image, un PDF, etc, la méthode POST sera également préférée.

En PHP, on peut donc récupérer les paramètres reçus depuis l'URL grâce à la variable super globale : `$_GET["email"]`. Super globale signifie qu'elle est accessible depuis n'importe où, quel que soit le scope (la portée).

Par exemple dans le cas de Youtube et du paramètre v :

```
echo $_GET["v"] ; // Affiche : US9JCsnAVTU
```

POST

La méthode POST est particulièrement utilisée pour envoyer des formulaires vers le serveur (ex : formulaire d'inscription). Son contenu est caché puisqu'il n'est pas dans l'URL de la requête mais dans le corps (body) de celle-ci (contrairement à GET, attention toutefois HTTP/HTTPS).

Elle permet également d'uploader des fichiers (images, PDF, word, ...) vers le serveur.

Les paramètres envoyés à POST sont accessibles en PHP via la variable super globale `$_POST["email"]`.

Formulaire

En html, les formulaires sont utilisés pour envoyer des données du client vers le serveur.

```
<form action="serveur.php" method="get">
  Prénom: <input type="text" name="name">
  <button type="button">Envoyer</button>
</form>
```

Dans cet exemple, on peut voir qu'« action » fait référence au fichier PHP qui va traiter le formulaire et la méthode que l'on souhaite utiliser.

En appuyant sur le bouton « Envoyer », le PHP va déclencher une requête GET vers la page `serveur.php`, équivalent à `/serveur.php ?name=prénom`, où prénom sera le prénom entré par l'utilisateur dans le formulaire.

Si la méthode du form est POST, la méthode POST sera employée. La requête vers le serveur sera `/serveur.php` sans paramètre. Le contenu du formulaire étant passé dans le corps de la requête POST.

La session

Par défaut, quand l'utilisateur communique avec le serveur, le protocole est dit stateless. C'est-à-dire que le serveur n'enregistre pas l'état d'une session de communication entre deux requêtes successives. La communication est formée de paires requête-réponse indépendantes et chaque paire requête-réponse est traitée comme une transaction indépendante, sans lien avec les requêtes précédentes ou suivantes. Néanmoins, il existe en PHP un mécanisme permettant de garder certaines informations à

propos de l'utilisateur (permettant par exemple de garder en mémoire son panier dans un site de commerce en ligne).

```
<?php
// Démarrage de la session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Variables des sessions
$_SESSION["myVar"] = "myValue";
?>

</body>
</html>
```

Pour utiliser la session, il faut que l'instruction `session_start()` soit la première instruction appelée. Il ne doit pas y avoir de code HTML avant cela, sinon cela ne fonctionnera pas. De la même manière qu'avec GET et POST, il faut utiliser la variable super globale `$_SESSION` pour avoir accès aux données de la session de l'utilisateur.

Pour supprimer la session (si l'utilisateur souhaite se déconnecter par exemple), il faut utiliser `session_unset()` pour supprimer les variables de la session de l'utilisateur. Pour détruire la session en cours, il faut utiliser `session_destroy()`.

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Supprimer toutes les variables de session
session_unset();

// Détruire la session
session_destroy();
?>
```



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

Un mécanisme similaire est utilisé pour les cookies :

https://www.w3schools.com/php/php_cookies.asp

Les exceptions

Un système de gestion d'exceptions permet de gérer les conditions exceptionnelles pendant l'exécution du programme. Lorsqu'une exception se produit, l'exécution normale du programme est interrompue et l'exception est traitée. Un bon exemple est la division par 0, qui provoque une erreur. En PHP, on écrira :

```
<?php
function divide($dividend, $divisor) {
    if($divisor == 0) {
        throw new Exception("Division by zero");
    }
    return $dividend / $divisor;
}

try {
    echo divide(5, 0);
} catch(Exception $e) {
    echo "Unable to divide.";
}
?>
```

On fait appel à la fonction divide où le diviseur est égal à 0. Avant de procéder au calcul, la fonction vérifie si le diviseur est égal à 0. Si c'est le cas, la fonction jette une exception avec comme message « Division by zero ». « Unable to divide » sera alors affiché à l'utilisateur. On aurait également pu afficher le message d'erreur de l'exception via \$e.getMessage().

Si on appelle la fonction divide avec un autre diviseur que 0, le résultat de la division sera alors affiché à l'utilisateur.

En savoir plus : https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_gestion_d%27exceptions

Les expressions régulières

Une expression régulière est une séquence de caractères qui forme un modèle de recherche. Lorsque vous recherchez des données dans un texte, vous pouvez utiliser ce modèle de recherche pour décrire ce que vous recherchez.

Plus d'infos :

https://www.w3schools.com/php/php_regex.asp

<https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql/916990-les-expressions-regulieres-partie-1-2>

Exercices - partie 2

Liste de courses

https://github.com/martini224/EFP_I01/tree/master/PHP/Liste_de_courses

Dans cet exercice, il faudra pouvoir créer sa liste de course à partir d'une liste d'articles prédéfinies.

L'objectif est que l'utilisateur puisse rentrer le nom d'un article via un formulaire. Pour être valide, le nom de l'article doit faire partie de la liste prédéfinie d'articles, triés par catégorie. Cette liste est reprise dans un fichier JSON disponible sur le repository GIT. Le code pour lire et transformer cette liste JSON en un tableau de Category est fourni dans le repository (pour les classes en PHP, voir partie 3 du cours).

L'ajout d'un article à la liste doit se faire via la méthode POST. Si le nom donné par l'utilisateur fait partie de la liste, il est ajouté dans la liste de l'utilisateur avec un message « article ajouté avec succès ». Sinon, l'article n'est pas ajouté et un message d'erreur « l'article n'existe pas » doit être affiché.

Pour pouvoir ajouter plusieurs articles sans que la liste ne soit remise à zéro à chaque ajout, il faudra utiliser une session et récupérer la liste d'articles encodée à chaque requête par l'utilisateur.

La liste des articles disponibles doit être affichée à l'utilisateur par catégorie. Par défaut, les articles d'une catégorie sont cachés, c'est quand on clique sur une catégorie que l'on peut voir les articles de celle-ci (<https://getbootstrap.com/docs/4.0/components/collapse/>). Cet affichage devra évidemment se faire en fonction des articles et catégories récupérées depuis le fichier JSON.

Une action permettant de supprimer la liste des articles de l'utilisateur doit exister.

L'utilisation du framework Bootstrap est demandé (code fournis).

Mettre un minimum de code dans index.php, et préférez un ou plusieurs fichiers séparés afin de garder votre code propre et lisible au maximum.

Le résultat devra ressembler à l'image ci-dessous :

<div>Liste mes articles</div> <div><ul style="list-style-type: none">• Pommes</div> <div>Supprimer tout</div> <div>Ajouter un article</div> <div>Nom de l'article : <input type="text"/> <input type="button" value="OK"/></div>	<div>Hors catégorie</div> <div>Fruits et Légumes</div> <div>Conserves</div> <div>Petit-déj / Goûter</div> <div>Bébé</div> <div>Crèmerie</div>
--	---

Aller plus loin : quand on encode plusieurs fois le même article, ne pas afficher plus d'une fois le nom, mais mettre entre parenthèse la quantité augmentée de 1 à chaque ajout.

Jeu de carte

https://github.com/martini224/EFP_I01/tree/master/PHP/Jeu_de_carte


Le but de l'exercice sera d'afficher une carte à jouer, et ce de 3 manières possibles :

- Une carte aléatoirement « piochée » dans un paquet de carte.
- Une carte dont on a envoyé le code en paramètre de la requête
 - o Le code d'une carte sera composé du numéro de la carte (2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A) avec le code de sa couleur (C pour trèfle, D pour carreau, H pour cœur, S pour pique). Par exemple le code de la carte 10 de cœur sera 10H.
- Deux cartes choisies aléatoirement, comparées l'une à l'autre. Par exemple As > 5.

Les images des cartes avec leur bon code est disponible sur le repository GIT.

Le résultat pourra ressembler à cela :

[Random card](#)
[Random cards - compare](#)
Display by code



Partie 3 – Pour aller plus loin

La programmation orientée objet

La programmation orientée objet (POO), ou programmation par objet, est un paradigme de programmation informatique. Il permet de représenter la réalité de manière plus ou moins abstraite. Ce concept de programmation permet donc de décrire et manipuler des objets et leurs propriétés, ainsi que les relations entre eux. Exemple : une table à 4 pieds, il y a 4 chaises autour de la table. Il permet également de décrire leurs comportements. Exemple : un chien peut aboyer.

Les classes

Une classe est un patron de conception à partir de laquelle on peut créer des objets que l'on peut manipuler.

Dans une classe, le vocabulaire utilisé est différent. Les variables d'une classe sont appelées attributs, et les fonctions sont des méthodes.

En savoir plus : <https://openclassrooms.com/fr/courses/2818931-programmez-en-orientee-objet-avec-c/2818941-introduction-a-la-programmation-orientee-objet>

```
class Player {
    private $name;
    private $level;

    function __construct ($name, $level) {
        $this->name = $name;
        $this->level = $level;
    }

    function displayName() {
        echo $this->name;
    }
}
```

Quand on crée un objet d'une certaine classe, on dit qu'on instancie la classe.

```
$object_player = new Player('joueur', 3);
```

Quand on affiche la variable via var_dump :

```
echo var_dump($object_player); // object(Player)#1 (2) {
["name":"Player":private]=> string(6) "joueur"
["level":"Player":private]=> int(3) }

$object_player->displayName(); // Fait appel à une méthode de l'objet
player
```

```
$object_player->name; // accède à l'attribut name qui renvoie le nom  
du joueur. On ne peut pas le faire en dehors de la classe car name est  
private dans la classe. Pour pouvoir y accéder, il faudrait que name  
soit public.
```

Pour en savoir plus sur les classes : <https://www.php.net/manual/fr/language.oop5.php>