Apprendre PHP et MySQL

Table des matières

Les bases

Un fichier **php** c'est avant tout un fichier HTML. La seul différence réside dans l'extension du fichier `.php` et non `.html`.

Afin de commencer à écrire du code php il faut ouvrir et fermer des balises php : `<?php ?>` ou `<? ?>`

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Les base de php</title>
</head>

<body>
    <h1>Les bases de php</h1>
    <!--
    Afin d'écrire du code PHP, il faut utiliser une balise
    un peu particulière : la balise php :
    -->
    <?php
    // Voici un commentaire en php
    /*
    Voici
    un commentaire
    sur
    plusieurs lignes
    */
```

```
    // PHP est un langage de programmation
    // basé sur un système d'instruction (d'ordre)
    // Chaque instruction doit se terminer par un ";"
    // Exemple : Afficher quelque chose en php :
    echo '<p>Affiché avec php</p>';
    ?>
</body>

</html>
```

Autre exemple :

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ecrire du php</title>
</head>

<body>
    <?php
    // Il est possible de placer du code php n'importe ou dans mon fichier
    // html :
    $nom = 'John Doe';
    $age = 20;
    ?>
    <h1>Bienvenue <?php echo $nom; ?></h1>

    <?php if ($age >= 18) { ?>
        <p>Hey vous êtes majeur !!</p>
    <?php  } else { ?>
        <p>Hey vous êtes mineur !!</p>
    <?php } ?>
</body>

</html>
```

Vous pouvez utiliser une syntax alternative graçe aux "short open tag" ainsi qu'a la "template syntax" de php :

- `<?php ?>` devient `<? ?>`
- `<?php echo ...; ?>` devient `<?= ?>`
- `if () {}` devient `<? if () : ?><? endif ?>`
- `for () {}` devient `<? for () : ?><? endfor ?>`
- `foreach () {}` devient `<? foreach () : ?><? endforeach ?>`
- `while () {}` devient `<? while () : ?><? endwhile ?>`
- etc ...

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ecrire du php</title>
</head>

<body>
    <?
    $nom = 'John Doe';
    $age = 20;
    ?>
    <h1>Bienvenue <?= $nom ?></h1>

    <? if ($age >= 18) : ?>
        <p>Hey vous êtes majeur !!</p>
    <? else : ?>
        <p>Hey vous êtes mineur !!</p>
    <? endif ?>
</body>

</html>
```

Configurer PHP

Php est un langage de programmation assez "ancien". Il a était conçu afin de créer des sites internet dynamique.

Il posséde un fichier de configuration avec tout un tas d'option activable ou desactivable afin d'assurer la sécurité et la bon déroulement de votre site internet.

Pour afficher ou connaitre la configuration de PHP nous devons utiliser l'instruction suivante :

```
phpinfo();
```

Afin de modifier la configuration de php il faut trouver ou localiser son fichier de configuration.

Pour cela, il suffit de regarder la séction : **Loaded Configuration File**

Les variables

Nous savons que PHP est un langage de programmation. Il possède un système d'instruction nous permettant de donner des ordres à notre ordinateur.

Nous avons l'instruction qui permet d'afficher : `echo`

```
echo 'Bonjour'; // Afficher la chaine de caractère "Bonjour"
echo 25; // Afficher l'entier 25
```

En php nous pouvons manipuler des données. Chaque donnée possède un type :

- `string` : Permet de manipuler du text
- `boolean` : Permet de manipuler "vrai" ou "faux"

- `int` : Permet de manipuler des chiffres sans virgule (non flottant)
- `float` : Permet de manipuler des chiffres à virgule

```
echo 'Test'; // Afficher la string "Test"
echo true; // Afficher le boolean "vrai"
echo 25; // Afficher le int 25
echo 34.6; // Afficher le float 34.6
```

Chaque donnée peut être enregistrer dans la mémoire de votre ordinateur. Cela nous permet de réutiliser, transformer et traiter nos données.

Pour cela php possède des variables :

```
$nom = "Jean"; // Ici on enregistre un "nom" dans la mémoire

// Grace au variables nous pouvons réutiliser autant de fois
// que l'on veut sa valeur
echo $nom;
```

PHP, comme tout les langages de programmation, est une véritable calculatrice :

Avec des "`int`" ou des "`float`" nous pouvons utiliser des opérations numéraires :

- "-" : Fais un soustraction
- "+" : Fais une addition
- "/" : Fais une division
- "%" : Fais le reste d'une division
- "*" : Fais une multiplication

```
$nombre1 = 10;
$nombre2 = 30;

echo $nombre1 - $nombre2;
```

PHP possède la possibilité de "concaténer" (assembler) des chaines de caractère (string).

Pour assemble des string, nous utilisons le "."

Par exemple :

```
echo 'Bonjour ' . $nom;
```

PHP possède aussi la possibilité de "interpoller" des chaines de caractères (string).

Grâce au guillemet double (""), php offre la possibilité d'interpréter des variables à l'intérieur de chaines de caractères

Exemple :

```
echo "Bonjour $nom";
```

Les conditions

Comme la plupart des langages de programmation, il est possible de réaliser des conditions. Ces conditions permettent d'exécuter du code on fonction de certaines données.

Exemple :

```
$age = 19;

if ($age >= 18) {
    echo 'Vous êtes majeur';
} elseif ($age < 6) {
    echo 'Vous êtes un enfant';
} else {
    echo 'Vous êtes mineur';
}
```

Les conditions utilisent des **expression boolean**. Ces dernières possède des opérateurs de comparaison :

- "==" : est égale a
- "!=" : n'est pas égale à
- "===" : est identique à
- "!==" : n'est pas identique à
- ">" : Est supérieur
- ">=" : Est supérieur ou égale
- "<" : Est inférieur
- "<=" : Est inférieur ou égale

```
if ($age == '19') {
    echo 'Vrai';
}

if ($age === '19') {
    echo 'Faux';
}
```

Toujours dans ces même expressions, nous pouvons utiliser des opérateur logique :

- "&&" : ET
- "||" : OU
- "!" : NON

```
if ($age >= 12 && $age <= 18) {
    echo 'Vous êtes adolescent';
}
```

En php, il existe une autre forme de "condition", c'est le "Switch".

Cette forme est un peu particulière, elle permet de tester les valeurs d'une variables et d'exécuter du code en conséquence.

Exemple :

```
$surname = "Johny";
```

```php
switch ($surname) {
    case 'Domi':
        echo 'Yo Domi !';
        break;
    case 'LouLou':
        echo 'Hello LouLou';
        break;
    case 'Johny':
        echo 'Johny !!!!!';
        break;
    default:
        echo 'Bonjour ' . $surname;
}
```

Les tableaux

En PHP, comme beaucoup d'autres langage de programmation, il éxiste une notion que l'on appel les tableaux (**array**).

Les tableaux permettent d'enregistrer plusieurs données de manière ordonné.

Il éxiste en PHP 2 sortes de tableaux :

- Les tableaux indéxé (indexed array)
- Les tableaux associatif (assosiativ array)

Les tableaux indéxé

Se sont des listes numéroté de 0 à X contenant de la données à chaque index.

Exemple :

```php
$notes = [12, 8, 16, 14, 7, 12];
```

Grace au tableaux il est possible d'accèder à un élément précis de la liste en utilisant son index

> **ATTENTION** Les index commence à partir de 0

```php
echo $notes[2]; // on affiche la note avec l'index 2 (note numéro 3) donc 16
```

Il est possible d'ajouter une valeur à un tableaux

Exemple, on ajoute la note 2

```php
$notes[] = 2;
array_push($notes, 2);
```

Il est possible de supprimer le dernier èlèment d'un tableaux

Exemple :

```php
array_pop($notes);
```

Il est possible d'ajouter une valeur au début du tableaux

Exemple :

```
array_unshift($notes, 10);
```

Il est possible de supprimer le premier élément d'un tableaux

Exemple :

```
array_shift($notes);
```

Les tableaux associatifs

Il éxiste des tableaux dit associatif.

Ils se comporte non plus comme des listes, mais plutôt comme un dictionnaire.

Se sont des tableaux qui ne sont plus numéroté mais associé à un clefs (key, ou mot de définition).

Exemple :

```
$eleve = [
'nom' => 'Doe',
'prenom' => 'John',
'age' => 17,
];
```

Nous pouvons facilement accéder à un élément précis de notre tableaux en demandant un clef :

```
echo $eleve['nom']; // Affiche : Doe
```

Nous pouvons ajouter un élément à notre tableaux

```
$eleve['classe'] = 'Terminal';
```

Il est possible de savoir si une clefs éxiste dans un tableaux associatif

```
array_key_exists('nom', $eleve); // Ici retourne : vrai (true)
array_key_exists('profPrincipal', $eleve); // Ici retourne : faux (false)
```

Il est possible de fusioner des tableaux

```
$notes2 = [5, 19, 18];

$notes3 = array_merge($notes, $notes2);
```

Travailler avec des tableaux peut être difficile. Afin de faciliter le travail, php met à disposition une fonction permettant d'afficher un tableaux à l'écran. Ainsi il est plus simple de savoir ce que contient notre tableaux

```php
print_r($notes3);
```

Très similaire au "**print_r**", il existe une fonction plus puissante :

```php
var_dump($eleve);
```

Il est possible de tester si un élément est présent dans un tableaux

```php
in_array(20, $notes); // Ici retourne : faux (false)
in_array(12, $notes); // Ici retourne : vrai (true)
```

Il est possible de compter le nombre d'éléments d'un tableaux grâce à la fonction count

```php
count($notes); // Ici retourne : 6
```

Les boucles

Les boucles permettent de répéter du code autant de fois que l'on désire.

Il éxiste en php 3 types de boucles :

- La boucle while
- La boucle for
- La boucle foreach

La boucle while

Cette boucle permet d'éxécuter du code autant de fois que nescessaire en fonction d'une condition !!

Exemple :

```php
$compteur = 0;

while ($compteur < 10) {
    echo "<p>Le compteur est a $compteur</p>";

    $compteur = $compteur + 1;
}
```

La boucle for

Cette boucle permet de créer une "itération".

Le principe est simple, exécuter du code autant de fois qu'on le désire

```php
for ($i = 0; $i < 10; $i++) {
    echo "<p>Je boucle $i de fois</p>";
}
```

La boucle foreach

PHP met à disposition une boucle spécialement conçu pour travailler avec des tableaux. Cette boucle permet de lancer du code sur chaque élément d'un tableaux. Cela fonctionne sur les tableaux indéxé (numéroté) mais sur les tableaux associatif.

Exemple :

```php
$notes = [12, 5, 8, 19, 16];


foreach ($notes as $note) {
    echo "<p>La note $note</p>";
}

foreach ($notes as $index => $note) {
    echo "<p>La note n°$index est $note</p>";
}

$eleve = [
    'nom' => 'Doe',
    'prenom' => 'John',
    'age' => 17,
    'classe' => 'Terminal',
];

foreach ($eleve as $clef => $valeur) {
    echo "<p>La clefs $clef de l'éléve contient $valeur</p>";
}
```

Les fonctions

Comme beaucoup d'autres langage de programmation, php possède un systèm de réutilisation de code : **les fonctions**. Ce sont des boites contenant du code isolé acceptant des **paramètres** et **retournant un valeur** :

```
Paramètres                Retour
   |                        |
   |                        |
1 ---> |---------------|
       |  additionner  | ---> 3
2 ---> |---------------|
```

En php il éxiste des milliers de fonctions permettant de faire diverse choses. Par éxemple, la fonction "echo". Cette fonction affiche quelque chose dans notre page ! Ou bien la fonction "array_pop" qui supprime le dernière élément d'un tableaux. L'intégralité des fonctions de php sont disponible sur le site de la documentation officiel :

[php.net : les fonction]

Les fonctions permettant de manipuler du texte

```php
// Retourne le nombre de character à l'intérieur de mon text
strlen('coucou'); // 6

// Permet de remplacer des bout de character dans du text
str_replace('copains', 'amis', 'Coucou les copains'); // Coucou les amis

// Il éxiste une fonction permettant de faire des concatenation.
// En utilisant cette fonction, les concatenation seront :
// - Bien plus performante
// - Sécurisé (on ne peus pas concatener n'importe quoi)
// - Très puissantes
sprintf('Coucou %s, vous avez %d ans', 'John', 32);
```

Les fonctions manipulant les dates

Cette fonction permet de récupérer la date du jour, elle accépte un paramètre : une chaîne de caractère spécifiant l'élément de la date que l'on veut récupérer :

- d (permet de récupérer le jour)
- Y (permet de récupérer l'année)
- m (permet de récupérer le mois)
- H (permet de récupérer l'heure)
- i (permet de récupérer les minutes)

```php
echo '<p>' . date('d-m-Y à H:i') . '</p>';
echo '<p>' . date('Y') . '</p>';
```

Créer ses propres fonctions

Il est possible de créer nos propre fonctions. Cela permet de réutiliser du code sans avoir à le copier / coller.

Généralement lorsque nous avons une opération un peu complexe, ou une opération qui se répéte dans notre code, il est conseillé de créer une fonction :).

Par exemple, essayons de créer une fonction qui additionne 2 nombres :

```php
function additionner(int $x, int $y): int
{
$resultat = $x + $y;

    return $resultat;

}

echo additionner('10', 4);
```

Nous pouvons typer nos paramètres et notre retour :

- int (entier)
- float (décimaux)

- string (text)
- array (tableaux)
- boolean (vrai ou faux)

Le résultat (le retour d'une fonction) peut-être récupérer dans une variable :

```php
$a = additionner(10, 4); // 14
$b = additionner($a, 2); // 16
$c = additionner($a, $b); // 14 + 16 = 30

echo $c; // Affiche : 30
```

Les inclusions

En php, il est possible d'inclure d'autre fichiers php. Une inclusion c'est un peu comme un copier / coller de fichier. Cela nous permet d'organiser notre site internet ainsi que de ne jamais répéter de code

Il faut pour cela utiliser la fonction include :

```php
include('mon-fichier.php');
```

Un exemple plus concret :

```php
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
crossorigin="anonymous">
    <title>Exemple d'inclusion</title>
</head>

<body>

    <!--
    L'en tête de ma page
    -->
    <? include('./includes/navbar.php') ?>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.min.js"
integrity="sha384-
IDwe1+LCz02ROU9k972gdyvl+AESN10+x7tBKgc9I5HFtuNz0wWnPclzo6p9vxnk"
crossorigin="anonymous"></script>
</body>

</html>
```

Les URL

Une url est composé de différent membre :

```
https://monsite.com/recette.php?id=5
  |           |          |        |
  |           |          |        |
protocole     |          |        |
              |          |        |
        nom de domaine   |        |
                      resource    |
                             Query String
```

Graçe au Query String, il est possible de passer des données de page en page.

Par exemple sur un page d'accueil, nous pourions générer le lien suivant :

> `/recette.php?id=5`

En cliquant sur ce lien, nous demandons à afficher la page `recette.php` et nous spécifions la recette que nous souhaitons afficher (id=5).

Les Query String sont formatté ainsi :

```
  ?id=5&email=john@mail.com
  |    |   |        |
  |    |   |        |
Demarre|  nom       |
  la   |          Valeur
 Query |
       |
  Sépare les
     query
```

Il est possible de récupérer ces paramètres dans un script php en utilisant la super global `$_GET`

Exemple, récupération de l'id passé en query string

```php
$id = $\_GET['id'];
```

> **ATTENTION !!!!**
>
> Les urls peuvent être modifié par un pirate méchant malveillant !
>
> - Donc il est déconseillé de transmettre des données sensible (exemple un mot de passe)
> - Rien ne nous assure la cohérence de la données dans votre script php (ex `/recette.php?id=<a href="">sdlfsd</a>`)
>
> C'est pourquoi il faut absolument tester et s'assurer que les données query string sont correct.

Il éxiste 2 fonctions php incontournable :

- `isset` : Permet de savoir si un paramètre éxiste

- `filter_var` : Permet de s'assurer que la données soit cohérente.

```php
// Imaginons que je soit sur la page : /recette.php?id=3
$id = $\_GET['id']; // $id = 3
```

```php
// Attention, si je suis sur la page : /recette.php
$id = $\_GET['id']; // PHP ERROR !!!!!!!
```

Pour éviter les erreur il suffit de tester l'éxistence de cette id :

```php
if (isset($_GET['id'])) { // vrai si l'id éxiste et faux sinon
    // on S'assure que l'id soit un entier,
    // ici soit la variable contiendra l'entier de l'id,
    // ou alors le boolean false
    $id = filter_var($\_GET['id'], FILTER_VALIDATE_INT);

    // Ici je sais que mon identifiant n'est pas correct
    if ($id === false) {
    }

} else {
// On pourrais renvoyer à l'accueil
// Ou alors afficher une page 404
// etc ....
}
```

La fonction filter_var est extrêmement puissante, elle permet de s'assurer de la cohérence de beaucoup de données :

Nous pouvons filtrer des entiers :

```php
filter_var('44', FILTER_VALIDATE_INT); // Retourne : int 44
filter_var('44.872', FILTER_VALIDATE_INT); // Retourne : int 44
filter_var('coucou les amis', FILTER_VALIDATE_INT); // Retourne : false
filter_var('67.827', FILTER_VALIDATE_FLOAT); // Retourne : float 67.827
```

Nous pouvons aussi valider des url et email

```php
filter_var('http://test.com', FILTER_VALIDATE_URL); // Retourne : string
http://test.com
filter_var('<p>ldkfhsdklfhsdl</p>', FILTER_VALIDATE_URL); // Retourne : false
filter_var('test', FILTER_VALIDATE_EMAIL); // Retourne : false
```

```php
filter_var('<p>Coucou les amis</p>', FILTER_SANITIZE_FULL_SPECIAL_CHARS); //
Retourne : string &amp;nbsp;p&amp;nbsp;etc....
```

les formulaires

Il est possible en php de récupérer et de traiter les données d'une formulaire HTML. Graçe au super global `$_GET`, `$_POST` et `$_FILE` nous avons à toutes les données d'un formulaire.

Afin d'envoyer des données à PHP en utilisant un formulaire HTML, il faut tout d'abord créer le formulaire.

Prenons un éxemple :

```
<form method="POST">
    <input type="email" value="john@mail.com" name="email" />
</form>
```

Un formulaire possède 2 atrributs :

1. La méthode HTTP (GET => obtenir, POST => créer). Si la méthode est GET alors les données du formulaires sont envoyé dans les query string de l'url Si la méthode est POST alors les données du formulaires sont envoyé dans le corps de la requête.
2. l'attribut action permet de determiné le fichier php qui recevra les données (lors du l'envoie du formulaire).

Nous pouvons récupérer les données d'un formulaire dans le script de l'action du formulaire en utilisant : - `$_GET` lorsque la méthode est GET - `$_POST` lorsque la méthode est POST

À l'intérieur de ces super global, nous pouvons récupérer la valeur d'un champs en utilisant son name

Exemple :

```
$_POST['email']; // récupére la valeur de l'input avec le name="email"
```

> **ATTENTION**
>
> Dans le cas on notre balise form ne possède pas d'action, c'est la page actuel qui sera utilisé pour recvoir les données.

Les sessions

Il éxiste en php, un moyen de concerver les données d'une page à une autre sans utilisé les données GET ou POST.

Ce moyen, c'est la SESSION. Un session correspond à un visiteur de votre site internet.

Afin d'utiliser les session en php, il faut absolument commencer notre script PHP par l'appel de la fonction :

```
session_start();
```

> **ATTENTION**
>
> Cette fonction "session_start()" doit ABSOLUMENT être la TOUTE PREMIERE LIGNE de votre script !!!

Graçe à cette fonction "session_start()", nous avons accés à une superglobal : $_SESSION.

```php
session_start();

// J'enregistre une information dans la session
$_SESSION['user'] = 'john@mail.com';

// Je récupére le user de la session :
$user = $_SESSION['user'];
```

Graçe au session nous pouvons conserver des informations entre nos différent scripts php (nos différentes pages).

> **ATTENTION**
>
> QUELQUES RÈGLES DE SÉCURITÉ :
>
> - Il est fortement déconseillé d'enregistrer autre chose qu'une données dite "scalaire" (boolean, int, float, string sont valide mais attention array, object, null ... ne sont pas valide !!)
> - n'enregistrer JAMAIS de HTML dans la session !!!!!!!!

Il est possible de "détruire" toutes les données enregistré dans la session :

```php
session_destroy();
```

Cette fonction est par exemple utilisé lors de la déconnexion.

Les cookies

Il éxiste une autre technique pour conserver des informations entre vos page que la SESSION.

Cette technique est plus puissante car l'on peut sécuriser et maitriser son éxistence.

Ce sont les COOKIES (pas les gateaux ...), des petits fichiers text enregistré sur votre navigateur et accessible en php.

Ces fichiers text contiennent différente informations :

```
- name : Le nom du fichier text
- valeur : Ce que contient le fichier text
- secure : Est-ce que le cookie est crypter
- httpOnly : Permet de rendre le cookie ilisible en javascript.
- expires : Détermine sa date d'expiration
```

La date d'expiration (expires) est exprimé graçe à un "timestamp". Un "timestamp" c'est le nombre de seconde
écoulé depuis le 1er janvier 1970.

Nous pouvons obtenir le timestamp actuel avec les fonction php "time()".

Il est donc possible de faire en sorte que notre cookie expire dans 24h :

```php
time() + 24 * 3600
```

Exemple un cookie qui éxpire un an plus tard :

```
time() + 365 * 24 * 3600
```

Pour créer un cookie en php, rien de plus simple :

```
setcookie('userEmail', 'john@mail.com', [
    'httpOnly' => true,
    'secure' => true,
    'expires' => time() + 90 * 24 * 3600,
]);

// Ainsi que :
$_COOKIE['userEmail'] = 'john@mail.com';
```

> **TRES TRES TRES IMPORTANT**
>
> 1. Les cookies ne peuvent enregistré QUE DES INFORMATIONS TEXTUELLE (string)
> 2. NE JAMAIS METTRE DE CODE HTML DANS LES COOKIES !!! (faille XSS)
> 3. Même si les cookies peuvent être créé avec PHP, rien n'empêche un méchant pirate de le modifier sur son navigateur !!!

Il est possible de récupérer à N IMPORTE QUEL MOMENT un cookie :

```
$email = $_COOKIE['userEmail']; // string john@mail.com
```

Il est possible de modifier un cookie en répétant la même opération que la création.

Pour supprimer un cookie, il suffit d'appeler "setcookie" et de lui spécifier une date d'expiration entérieur à aujourd'huie :

```
setcookie('userEmail', '', [
'expires' => 1,
]);
```

Attention, le fais de supprimer un cookie avec setcookie ne supprime pas le cookie dans la superglobal :

```
$_COOKIE['userEmail']; // string john@mail.com
```

Pour totalement supprimer un cookie, il faut aussi "unset" le cookie lui même :

```
unset($_COOKIE['userEmail']);
```

Les bases de données

PHP est souvent associé à une base de données. La plus célèbre est "MySQL". Une base de données c'est comme un tableur excel, c'est une grande armoire contenant des tables. Ces tables contiennent des colonnes (les champs) et des lignes (les entrées).

Afin d'utiliser une base de données en PHP il faut tout d'abord se connécter. Pour manipuler des bases de données PHP utilise un « objet » : PDO.

```php
// Nous nous connéctons à notre base de données
$connection = new PDO(
    'mysql:host=db;dbname=cookme;charset=utf8',
    'root',
    'root',
);
```

Graçe au langage SQL, il est possible de récupérer des données de notre base de données :

Exemple séléctioner tout les users :

```sql
SELECT * FROM users;
```

Et bien en php, graçe à PDO nous pouvons facilement lancer des requêtes et récuperer les résultat :

```php
// La première étape est de préparer la requête
$requete = $connection->prepare('SELECT * FROM users');

// La seconde étape est de lancer la requête
$requete->execute();

// On transforme notre résultat en tableaux php
$users = $requete->fetchAll();
```

Nous récupérons un tableaux contenant chaque utilisateur sous forme de tableaux. C'est don un tableaux de tableaux

Exemple afficher une balise h2 avec l'email de chaque utilisateur

```php
foreach ($users as $user) {
    echo '<h3>' . $user['email'] . '</h3>';
}
```

En php, graçe à mysql, nous pouvons aussi insérer de nouveau résultat en utilisant les fonctions précédentes et la requête SQL suivante :

```sql
INSERT INTO users (firstname, lastname, email, password, sexe, birthdate)
VALUES ('test', 'test', 'test@mail.com', 'test', 'homme', '1990-03-21')
```

```php
// On prépare la requête. Pour les requêtes sur plusieur vous
// pouvez utiliser les chaines de caractère HEREDOC :
$requete = $connection->prepare(<<<SQL
    INSERT INTO users (firstname, lastname, email, imageUrl, password, sexe,
birthdate)
    VALUES ('test', 'test', 'test@mail.com', 'https://external-
content.duckduckgo.com/iu/?u=http%3A%2F%2Ffree-profile-pics.com%2Fprofile-
pictures%2F01232014%2Fdownload%2Fmr-bean-profile-picture-
180x180.png&f=1&nofb=1&ipt=bedc69dcbd46ceccc906b9ab7a3bb57641a67c7695f3acd1550074
83280df0c4&ipo=images', 'test', 'homme', '1990-03-21')
SQL);

// Il suffit d'éxécuter la requête pour lancer l'insertion :
$requete->execute();
```

En régle générale, les données que nous insérons dans la base de données viennent d'un formulaire.

Cela implique que les données sont stocké dans des variables.

Afin d'éviter **TOUT PROBLEME DE SECURITE**, PDO à créé un système d'insertion de variable très puissant :

```php
// immaginons les variables suivantes :
$firstname = 'Jack';
$lastname = 'Doe';
$email = 'jack@mail.com';
$imageUrl = 'https://external-content.duckduckgo.com/iu/?u=http%3A%2F%2Ffree-
profile-pics.com%2Fprofile-pictures%2F01262014%2Fdownload%2Fwolf-profile-picture-
180x180.png&f=1&nofb=1&ipt=b8ff28ae15e0aa213a1e476a12a0be21ec4677c83d021e5cb9a03c
a339e572eb&ipo=images';
$password = 'jack';
$sexe = 'homme';
$birthdate = '1992-11-10';
```

Afin d'insérer la moindre variable dans une base de données, il est obligatoire d'utiliser des paramètre SQL :

```php
$requete = $connection->prepare(<<<SQL
    INSERT INTO users (firstname, lastname, email, imageUrl, password, sexe,
birthdate)
    VALUES (?, ?, ?, ?, ?, ?)
SQL);
```

Afin d'éviter l'injéction SQL, nous allons pouvoir passer les variables dans un tableaux lors de execute :

```php
$request->execute([
    $firstname,
    $lastname,
    $email,
    $imageUrl,
    $password,
    $sexe,
    $birthdate,
]);
```

Vous avez aussi la possibilité de remplacer les "?" par des paramètres nommées :

```php
$requete = $connection->prepare(<<<SQL
    INSERT INTO users (firstname, lastname, email, imageUrl, password, sexe,
birthdate)
    VALUES (:firstname, :lastname, :email, :imageUrl, :password, :sexe,
:birthdate)
SQL);
```

Afin d'éviter l'injéction SQL, nous allons pouvoir passer les variables dans un tableaux lors de execute :

```php
$request->execute([
    'firstname' => $firstname,
    'lastname' => $lastname,
    'email' => $email,
    'imageUrl' => $imageUrl,
    'password' => $password,
    'sexe' => $sexe,
    'birthdate' => $birthdate,
]);
```

> **ATTENTION** :
>
> Lorsque l'on demande des SELECT et que l'on doit passer une variable dans notre SQL, nous devons aussi utiliser les SQL :
>
> ```php
> $limit = 2;
>
> $requete = $connection->prepare('SELECT * FROM users LIMIT ?');
> $requete->execute([$limit]);
> $users2 = $request->fetchAll();
> ```