

# PHP

développer un site web avec une base de données

C'est parti →

# Programme

1. Introduction à PHP
2. Présentation du langage
3. Bonnes pratiques
4. Fichiers
5. Persistance de données de contexte
6. Bases de données

# Votre formateur

Benoît Tavernier 😊

- Expert PHP Symfony
- Certifié PHP, Linux admin
- Formateur depuis 2010
- Responsable pédagogique pendant 7 ans
- 5 ans chez SensioLabs, créateur de Symfony



# Introduction à PHP

# Introduction à PHP

1. Qu'est-ce que PHP ?
2. Contexte Web
3. Contexte CLI
  1. Environnement de travail
  2. Serveurs Web
4. Serveurs de gestion de bases de données
1. Des questions ?

# Qu'est-ce que PHP ?

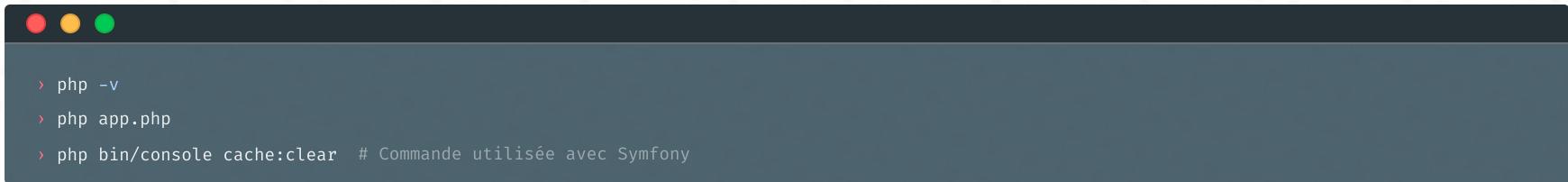
- Langage de programmation open source, licence PHP
- Développé par une Core Team
- Interpréteur écrit en C
- Organisé avec un noyau et des extensions natives ou tierces
- Zend Engine : Compilateur et environnement d'exécution du code PHP
- Interfaces (SAPI) : *CLI, Apache, phpdbg, CGI, FPM, Embed*
- Projets notables : *WordPress, Drupal, PrestaShop, Magento, Symfony, Laravel...*

# Contexte Web

- Un serveur Web lance l'application PHP à chaque appel
- PHP utilisé pour déterminer une réponse HTTP statique au client (un navigateur web)
- Calcul effectué côté serveur (inaccessible par le client)

# Contexte CLI

- Pratique pour du debug



A screenshot of a macOS terminal window. The window has a dark blue background and a dark grey header bar with three colored circular icons (red, yellow, green) on the left. The terminal shows the following command history:

```
> php -v
> php app.php
> php bin/console cache:clear # Commande utilisée avec Symfony
```

# Environnement de travail

- Terminal : *Cmd, iTerm, Cmder, Git Bash...*
- Éditeur de code : *VSCode, PHPStorm...*
- Navigateur : *Chrome, Firefox, Safari...*
- Serveur Web complet (HTTP, PHP, MariaDB)

# Serveurs Web

- Au choix :
  - PHP built-in server
  - Serveurs HTTP : *Apache, NGINX, FrankenPHP...*
  - Pack avec Serveur local, PHP et bases de données : *Laragon, XAMPP...*
  - Solutions Cloud
  - Docker

Pour la formation, vous utiliserez XAMPP qui fournit : Apache, PHP, MariaDB, phpMyAdmin

Attention aux versions de PHP fournies !

# Serveurs de gestion de bases de données

- Acronymes utilisés : SGBD  ou DBMS 
- Serveurs SQL et NoSQL
- Solutions SQL :
  - Oracle
  - PostgreSQL
  - SQL Server
  - MariaDB
  - MySQL
  - SQLite
  - ...
- Interfaces de gestion telles que phpMyAdmin à installer en complément 😊

# Pratique

TP : initialisation d'un projet Web

En PHP, une fonction `phpinfo()` permet d'obtenir toutes les informations concernant votre configuration PHP actuelle.

1. Créer le dossier `formation` de notre application web dans le dossier

`C:\laragon\www` de `Laragon`

2. Dans ce dossier, créer un fichier `index.php` contenant ceci :

```
<?php  
phpinfo();
```

3. Visualisez le résultat à travers un navigateur web sur `http://formation`



 15 mn

# Pratique

TP : Vérification de version de PHP

1. Essayez déterminer la version de PHP disponible sur :
  1. votre environnement Web ;
  2. votre environnement en console.



 15 mn

Des questions ?

\* \* \*



## Présentation du langage

# Présentation du langage

1. Fichiers PHP
2. Utilisation dans une page HTML
3. Types primitifs
4. Variables et constantes
5. Fonctions
6. Architecture
7. Inclusions de fichiers
8. Contexte et données du serveur
9. Structures de contrôle

# Fichiers PHP

- Extension de fichiers `.php` afin d'être interprétés par PHP
- Regardez aussi l'accès aux fichiers : <http://localhost/dossier/fichier.php>

# Utilisation dans une page HTML

- Code PHP à marquer entre 2 balises :

php

```
<section>
    <h1>Ce fichier contient directement du HTML et du PHP</h1>
    <p>Ce fichier ne donnera au client HTTP qu'un résultat HTML. (HTML par défaut)</p>
    <?php echo 'PHP peut faire des calculs et retourner du <span style="color: #e5989b;">HTML</span>'; ?>
</section>
```

- Quand un fichier ne contient que du code PHP :

php

```
<?php
echo 'on ne ferme pas la balise PHP, mais on a un retour à la ligne 😊 ';
```

# Types primitifs

- scalaires :

```
var_dump(true, false);           // bool
var_dump(-5, 0, 10);            // int
var_dump(0.2, .5);              // float
var_dump('bravo');              // string
```

php

- composites :

```
var_dump([1, 1, 3], ['label' => 'cheese', 'price' => 3.51]); // array
var_dump(new stdClass());          // object
```

php

- spéciaux :

```
var_dump($trucInconnu);          // null
var_dump(fopen('file', 'r'));
```

php

- pseudo-types : `iterable`, `void`, `callable` (notamment)

# Variables et constantes

- En PHP, pour désigner une variable, on la précède **toujours** par le symbole `$` :

```
$count = 0;  
$count = $count + 1;  
echo $count;
```

php

- En PHP, les constantes n'ont pas de préfixe, mais une **convention** d'écriture `UPPER_CASE` :

```
// Syntaxe 1 :  
define('MAX_ITEMS', 10);  
  
// Syntaxe 2 :  
const MAX_ITEMS = 10;      // Syntaxe recommandée  
echo MAX_ITEMS;
```

php

# Tableaux

php

```
// On commence par un tableau numérique,  
$ids = [1, 2, 5, 10, 12];  
var_dump($ids);  
  
// et on continue par un tableau associatif  
$book = [  
    'id' => 1,  
    'isbn' => '0-575-04463-2',  
    'title' => 'Pyramides',  
    'publicationDate' => new DateTime('15. June 1989'),  
    'available' => true,  
    'tags' => ['fantasy', 'novel'], // Cette virgule en fin de liste est OK 😊  
];  
  
var_dump($book['publicationDate']);
```

# Fonctions

php

```
$x = 5;

function multiply(float $x, float $y): float {
    return $x * $y;
}

$result = multiply(1.5, 2);

// $result -> float: 3
// $x      -> float: 5
```

# Fonctions variadiques

php

```
function multiply(float ...$numbers): float {
    $result = 1;

    foreach($numbers as $number) {
        $result *= $number;
    }

    return $result;
}

$result = multiply(1.5, 2, 4);
```

# Closures et références

php

```
$x = -5;
$debug = true;

$closure = function (float &$parameterGivenOnCall) use ($debug): void {
    if ($debug === true && $parameterGivenOnCall <= 0) {
        echo 'Le nombre transmis est inférieur ou égal à 0';
    }

    $parameterGivenOnCall = $parameterGivenOnCall > 0 ? $parameterGivenOnCall : 0.;
};

function applyClosureOnNumber(float &$number, callable $callback): void {
    $callback($number);
}

applyClosureOnNumber($x, $closure);

var_dump($x); // float: 0
```

# Architecture

- Idée d'architecture

```
project/
  └── app/
    ├── entity/
    ├── services/
    ├── controllers/
    │   └── homepage.php
    └── templates/
        └── homepage.php
  └── config/
  └── public/          # Ressources publiques, accessibles depuis une URL
    ├── index.php
    └── assets/
      ├── images/
      └── css/
          └── global.css
  └── vendor/          # Librairies tierces
  └── phpunit.xml.dist
  └── composer.lock
  └── composer.json
```

# Inclusions de fichiers

- PHP propose 2 instructions d'inclusion de fichiers qui fusionnent les contextes :

```
require_once './some_directory/file.php'; // Plutôt pour des librairies
require 'file.php'; // Plutôt pour des blocs à inclure
echo $x; // $x est probablement défini dans 1 des fichiers inclus
```

php

```
<?php

// On cherche à inclure l'équivalent de ../lib/database.php en chemin absolu
require_once dirname(__DIR__) . '/lib/database.php';
require_once dirname(__DIR__) . '/config/database.php';

$dbConnection = dbConnect(DB_PARAMS);
// [...]
require dirname(__DIR__) . '/templates/dashboard.php';
```

php

Note : Pour éviter des bugs, il est préférable de définir des chemins absolus pour les fichiers à inclure.

# Inclusion de fichiers - configuration

- Cas particulier assez utile :

```
// data/plants.php
```

```
return [
    [
        'id' => 1,
        'name' => 'Paulownia tomentosa',
    ],
    [
        'id' => 2,
        'name' => 'Phyllostachys nigra',
    ]
];
```

php

```
// app.php
```

```
$plants = require 'data/plants.php';
```

php

# Contexte et données du serveur

- Variables superglobales :
  - Variables d'environnement : `$_ENV` ou `getenv()`
  - Variables du contexte HTTP : `$_GET` , `$_POST`
  - Variables du contexte PHP : `$_SESSION` , `$_COOKIE`
- Constantes du langage : `__DIR__` , `__LINE__` , `__FUNCTION__`

J'ai placé des liens pour avoir une liste exhaustive de ces variables et constantes, mais attention de ne pas vous blesser avec, ça parle également de *classes* et *namespaces* 😊

# Structures de contrôle

- tests :

1. `if` et opérateur ternaire
2. `switch` et `match`

- boucles :

1. `foreach`
2. `for`
3. `while`

# Opérateurs de comparaison

php

```
$a == $b
$a != $b
$a === $b
$a !== $b
$a < $b
$a <= $b
$a > $b
$a >= $b
is_bool($a), is_numeric($a)...
!is_bool($a), !is_numeric($a)...
$a <=> $b // int: <0 | 0 | >0
```

# Structures de contrôle - if

php

```
if ($age < 18) {           // Les accolades définissent un bloc multiligne
    echo 'Salut';
} elseif ($age < 50) {     // "elseif" ou "else if"
    echo 'Bonjour';
} else {
    echo 'Votre Seigneurie';
}
```

- Opérateurs ternaires en guise de bonus 😊

php

```
$array = is_numeric($ids) ? [$ids] : $ids;

# + short syntax
$title = $label ?: $defaultLabel;      // si $label est null, alors $defaultLabel

# + null coalesce
$title = $label ?? $defaultLabel;      // si $label est null ou indéfini, alors $defaultLabel
```

Voir aussi : <https://www.php.net/manual/en/control-structures.if.php>

# Structures de contrôle - `switch`

```
switch ($debugLevel) {  
    case LEVEL_DEBUG:  
        echo "L'application va faire toutes sortes de choses pour analyser la situation";  
        break;  
    case LEVEL_CRITICAL:  
    case LEVEL_EMERGENCY:  
        echo "L'application craque nerveusement";  
        break;  
    default:  
        echo "All is fine 🔥";  
}
```

php

```
$location = match($mode) {  
    'relative' => $url,  
    'absolute' => customFunction($url),  
}
```

php

Voir aussi : <https://www.php.net/manual/en/control-structures.match.php>

# Structures de contrôle - **foreach**

php

```
$ids = [1, 5, 10];
foreach ($ids as $id) {
    echo $id;
}
```

php

```
$addresses = [
    [
        'label' => 'main',
        'location' => ['street' => '1 avenue du Maine', 'postalCode' => '75014', 'city' => 'Paris']
    ],
    [
        'label' => 'work',
        'location' => ['street' => '3-5 Rue Maurice Ravel', 'postalCode' => '92300', 'city' => 'Levallois-Perret']
    ],
];

foreach ($addresses as $address) {
    echo '<li>' . $address['label'] . '</li>';
}
```

# Structures de contrôle - **for**

- Instruction avec condition libre
- N'implémente pas la forme ~~for .. in ..~~

php

```
for ($i = 0; $i < 10; $i++) {  
    echo "$i ";  
}
```

# Structures de contrôle - `while`

```
$handle = fopen("test.csv", "r"));

if (($handle !== false) {
    while (($data = fgetcsv($handle, 1000, ",")) !== false) {
        $num = count($data);

        for ($c=0; $c < $num; $c++) {
            echo $data[$c] . "<br />\n";
        }
    }
    fclose($handle);
}
```

php

# Pratique

TP : Architecture de projet

→ *Créer une architecture de projet réaliste*

Réaliser l'architecture proposée un peu plus tôt

Penser à comment atteindre telle ou telle page web de votre application en appelant toujours `index.php` par votre serveur.



⌚ 40 mn

# Pratique

TP : Administration et liste d'utilisateurs

→ *Créer une page d'affichage des utilisateurs*

1. Créer un fichier PHP avec une liste d'utilisateurs avec pour chacun :

- id
- firstName
- creationDate
- roles : liste de roles ('ROLE\_USER', 'ROLE\_ADMIN'...)
- username



 15 mn

2. Dans une page dédiée de votre application, inclure ces définitions

3. Afficher proprement la liste des utilisateurs 😊

# Pratique

TP : Formulaire de recherche

→ Chercher un utilisateur d'après un critère saisi par formulaire

1. Créer une page avec un formulaire HTML pour demander un `username`

2. Si le formulaire est bien transmis en méthode `POST` :

1. Récupérer le `username`

2. Rechercher l' `id` de l'utilisateur

3. Si l'utilisateur est trouvé : afficher la page de l'utilisateur

4. Si l'utilisateur n'est pas trouvé : afficher une erreur



 15 mn

Des questions ?

\* \* \*



## Bonnes pratiques

# Bonnes pratiques

1. Recommandations PSR

2. Standards de codage

3. Méfiez-vous des utilisateurs

4. Responsabilité

5. Architecture

6. Code

7. Debug

# Recommandations PSR

- Recherche de consensus sur des implémentations récurrentes en PHP
- Groupe de travail PHP FIG
  - Voir les membres actuels
- Documents importants :
  - [PSR-4](#) : Autoloader
  - [PSR-3](#) : Logs
  - Standards de codage

# Standards de codage

- Conventions :
  - [PSR-1](#)
  - [PSR-2](#) ↗ étend et remplace PSR-1
  - [PSR-12](#) ↗ étend et remplace PSR-2
  - [PER coding style](#) ↗ étend et remplace PSR-12
- Outils utiles :
  - [PHP\\_CodeSniffer](#) ↗ : linter et fixer
  - [PHP Coding Standards Fixer](#) ↗ : linter et fixer

# Méfiez-vous des utilisateurs

- Vérifiez toutes les données entrantes !

```
$id = filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT);
```

php

# Responsabilité

- Ne codez pas tout vous-mêmes 😬
- Un jour, visitez Packagist
- Mettez à jour vos librairies tierces !

Nous ferons 1 ou 2 exemples dans ce cours

# Architecture

- Separation of concerns
- MVC

# Code

- DRY, Don't Repeat Yourself
- KISS, Keep it simple, Stupid!

# Debug

- Utilisation de Xdebug
- Utilisation de librairies PHP

# Xdebug

- Documentation officielle

Utilité :

- Décoration de `var_dump()`, affichage de la pile d'appels.
- Profiling
- Tracing
- Debugging

# Activation du debug

Debug:

1. Déclencher Xdebug :

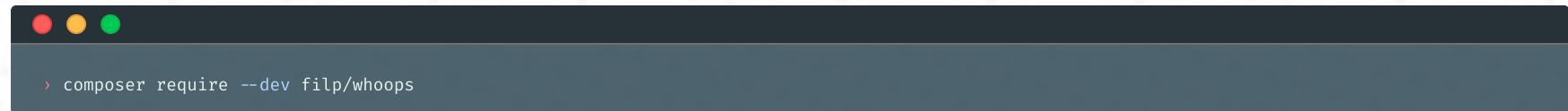
- Extension de navigateur web
- Variable d'environnement

2. Écouter Xdebug :

- Bouton "Start Listening for PHP Debug Connections" dans PHPStorm

3. Aller sur la page web du projet PHP 

# Outils de debug : Whoops



```
composer require --dev filip/whoops
```

php

```
$whoops = new \Whoops\Run();
$whoops->pushHandler(new \Whoops\Handler\PrettyPageHandler());
$whoops->register();
```

→ Essayez de faire une erreur dans le code 🧑

# Outils de debug : Dump



```
▶ composer require --dev symfony/var-dumper
```

→ Essayez de faire un `dump(server: $_SERVER)`

## Outils de debug : Dump server (bonus)

php

```
$cloner = new VarCloner();
$fallbackDumper = \in_array(\PHP_SAPI, ['cli', 'phpdbg']) ? new CliDumper() : new HtmlDumper();
$dumper = new ServerDumper('tcp://127.0.0.1:9912', $fallbackDumper, [
    'cli' => new CliContextProvider(),
    'source' => new SourceContextProvider(),
]);
VarDumper::setHandler(function (mixed $var) use ($cloner, $dumper): ?string {
    return $dumper->dump($cloner->cloneVar($var));
});
```



A screenshot of a terminal window with a dark background and light-colored text. The window has three circular icons in the top-left corner (red, yellow, green). The text in the terminal shows the command `./vendor/bin/dump-server`.

→ Essayez de faire un `dump(server: $_SERVER)`

Des questions ?

\* \* \*



# Fichiers

# Fichiers

1. Comment lire un fichier
2. Comment écrire un fichier
3. Gestion du système de fichiers
4. Informations

# Comment lire un fichier

- Lecture complète : `file_get_contents()`, `file()`, `readfile`
- Lecture séquentielle : `fread()`, `fgets()`, `fscanf()`
- En bonus, pour les fichiers CSV, séquentiellement : `fgetcsv()`

# Comment écrire un fichier

- `file_put_contents()` ↗
- `fputs()` ↗ / `fwrite()` ↗
- `fputcsv()` ↗

# Gestion du système de fichiers

- `mkdir()` ↗, `rmdir()` ↗, `chmod()` ↗, `chown()` ↗
- `copy()` ↗, `rename()` ↗, `unlink()` ↗
- Fonction particulièrement utile pour des formulaires : `move_uploaded_file()` ↗

# Informations

- Liste des fichiers : `glob`
- `file_exists()`
- `is_dir()`, `is_link()`, `is_file` ...
- `dirname()`, `basename()`, `pathinfo()`

# Pratique

TP : Fichiers [CSV](#)

→ Déclarer des produits dans un fichier CSV

1. Créer un fichier [products.csv](#) avec pour chaque produit :

- id
- name
- picture
- price

2. Créer un service (une fonction) permettant de récupérer ces produits

3. Créer une page web pour afficher ces produits



 30 mn

# Pratique

TP : Fichiers [YAML](#)

→ Déclarer des produits dans un fichier YAML

1. Créer un fichier [products.yaml](#) avec pour chaque produit :

- id
- name
- picture
- price

2. Chercher une librairie pour lire ce fichier [yaml](#)

3. Tester la librairie 😊



15 mn

# Pratique

TP : Fichiers de logs

→ *Disposer d'un service d'écriture de logs*

1. Créer une fonction d'écriture de logs dans un fichier dédié
2. Importer cette définition dans votre `index.php`
3. Écrire un log depuis une de vos pages Web



 15 mn

Des questions ?

\* \* \*



## Persistante de données de contexte

# Persistance de données de contexte

1. Session

2. Cookies

# Session

- Stockage de données sur le serveur
- Identifiant de session enregistré comme `cookie`, nommé par défaut `PHPSESSID`
- Restauration de ces données dans `$_SESSION` avec `session_start()`
- `$_SESSION` est une superglobale qui réagit comme un simple tableau

php

```
session_start();  
  
$user = $_SESSION['user'] ?? null;  
  
// déconnexion de l'utilisateur :  
unset($_SESSION['user']);
```

`session_start()` doit absolument être appelé avant tout contenu de réponse HTTP  
(la fonction modifie la `response` HTTP pour y placer un cookie )

# Cookies

- Valeurs des cookies dans `$_COOKIE`
- écriture d'un cookie avec `setcookie()` ↗

# Pratique

TP : Authentification

→ Mémoriser l'authentification d'un utilisateur en session PHP

1. Créer une page de connexion avec un formulaire contenant :

- username
- password

2. Utiliser `password_hash()` pour sécuriser les mots de passe.

(choisir `argon2id` si possible)

3. Stocker l'utilisateur authentifié en session PHP

4. Tester l'authentification et le rôle de l'utilisateur sur une autre page web



40 mn

# Pratique

TP : Déconnexion

→ Supprimer l'authentification d'un utilisateur en session PHP

1. Créer une page de déconnexion
2. Supprimer l'authentification de la session si elle existe
3. Rediriger l'utilisateur vers la page d'accueil du site



 10 mn

# Pratique

TP : Autorisation

→ Vérifier les droits utilisateur afin d'accéder à une page Web

1. Vérifier si l'utilisateur est connecté
2. Vérifier si l'utilisateur possède un `role` spécifique
3. Rediriger l'utilisateur si nécessaire
4. Essayer d'utiliser des `services` pour faire les vérifications précédentes 😊



⌚ 20 mn

Des questions ?

\* \* \*



## Bases de données

# Bases de données

1. MariaDB
2. Créer une table
3. Stocker des données
4. Récupérer des données
5. Aggrégations de données
6. Connexion avec PHP
7. Requêtes avec PDO
8. Manipulation de données avec PDO

# MariaDB

- MariaDB prend la relève de MySQL suite au rachat de ce dernier par Oracle
- Compatible MySQL
- Projet open source
- <https://mariadb.org/>
- Nombreux clients disponibles : `mysql` en terminal, [phpMyAdmin](#) ...

# Pratique

TP : Créer une base de données

→ Accéder au serveur et paramétrer une base

1. Accéder au choix à votre serveur MariaDB via :

- `phpMyAdmin` (Web)
- `mysql` (CLI)

2. Créez une base de données avec un encodage `utf8mb4` par défaut

```
CREATE DATABASE training CHARACTER SET utf8mb4;
```

mysql



10 mn

# Créer une table

mysql

```
CREATE TABLE advert
(
    id          INT UNSIGNED NOT NULL AUTO_INCREMENT,
    title       VARCHAR(50) NOT NULL,
    comment     TEXT,
    publication_date DATETIME,
    creation_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (id)
);
```

```
SHOW TABLES;
```

```
ALTER TABLE advert MODIFY comment VARCHAR (255) NOT NULL;
```

```
SHOW CREATE TABLE advert \G
```

Voir aussi : <https://mariadb.com/kb/en/data-types/>

# Stocker des données

mysql

```
INSERT INTO advert (`title`, `comment`, `publication_date`, `unpublication_date`)
VALUES ('aa', 'aaaa', NOW(), DATE_ADD(NOW(), INTERVAL 7 days)),
       ('bb', 'bbbb', NOW(), DATE_ADD(NOW(), INTERVAL 30 days));

SELECT * FROM advert \G
```

# Récupérer des données

mysql

```
SELECT * FROM advert ORDER BY `title` DESC;  
SELECT * FROM advert WHERE `id` = 1;
```

Voir la documentation officielle

Voir aussi : <https://mariadb.com/kb/en/built-in-functions/>

# Aggrégations de données

mysql

```
SELECT COUNT(*) AS nb_lines FROM advert;
SELECT COUNT(publication_date) FROM advert;

SELECT COUNT(IF(unpublication_date IS NOT NULL AND unpublication_date > NOW(), 1, NULL)) AS nb_active_ads
FROM advert;
```

mysql

```
SELECT GROUP_CONCAT(`title`) AS `adverts_published_by_month`, EXTRACT(YEAR_MONTH FROM `publication_date`) AS `month`
FROM `advert`
GROUP BY `month`;
```

mysql

```
SELECT COUNT(*) as `nb_adverts`, EXTRACT(YEAR_MONTH FROM `publication_date`) AS `month`
FROM `advert`
GROUP BY `month`
HAVING `nb_adverts` > 0;
```

# Connexion avec PHP

2 interfaces disponibles :

- MySQLi
- PDO (connecteur pour MySQL, PostgreSQL, SQLite...)

# Connexion avec MySQLi

php

```
$connexion = mysqli_connect('localhost', 'username', 'password', 'database_name');

if ($connexion === false) {
    echo 'Unable to reach the database: ' . mysqli_connect_error();
    exit();
}
```

php

```
$connexion = new mysqli('localhost', 'some_username', 'some_password', 'database_name');

if (mysqli_connect_errno()) {
    echo 'Unable to reach the database: ' . mysqli_connect_error();
    exit();
}
```

## Connexion avec PDO

php

```
// Au choix :  
$dsn = 'mysql:host=localhost;dbname=some_database;charset=utf8mb4';  
$dsn = 'sqlite:/path/to/database.sqlite3';  
$dsn = 'pgsql:host=localhost;dbname=some_database;charset=utf8mb4';  
$dsn = 'sqlsrv:Server=localhost;Database=some_database;charset=utf8mb4';  
  
try {  
    $db = new PDO($dsn, 'some_username', 'some_password', [  
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,  
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,  
    ]);  
} catch(PDOException $pe) {  
    echo 'Unable to reach the database: ' . $pe->getMessage();  
}
```

# Requêtes avec PDO

php

```
$statement = $pdo->prepare('SELECT * FROM book WHERE id = :id');

$statement->execute([
    'id' => 1,
]);

$row = $statement->fetch();
// Voir aussi : $rows = $statement->fetchAll();

if ($row !== false) {
    // ...
}
```

# Manipulation de données avec PDO

php

```
$statement = $pdo->prepare('INSERT INTO book (title) VALUES (:title)');
$statement->execute([
    'title' => 'Le Huitième Sortilège',
]);

$statement = $pdo->prepare('DELETE FROM book WHERE id = :id');
$statement->execute([
    'id' => 1,
]);
```

Des questions ?

\* \* \*

# Table des matières

1. Introduction à PHP
2. Présentation du langage
3. Bonnes pratiques
4. Fichiers
5. Persistance de données de contexte
6. Bases de données

Merci 