PHP - POO & PDO

DELORD Clément 06 68 12 81 21 cdelord@dawan.fr

Liens utiles

Conférence

> https://bbb.dawan.fr/b/cle-hci-tyj-dqt

Doc PHP

> https://php.net

Github

> https://github.com/DELORD-C/PHP-POO-PDO

MonCompte Dawan

> https://moncompte.dawan.fr

TheMovieDbApi

> https://developers.themoviedb.org/3/getting-started/authentication Clé d'API : 625b3e1220c0fca7c7ac7f6fcca786ac

Composer

> https://getcomposer.org/

Bootstrap

> https://getbootstrap.com/docs/5.2/ ```shell= composer require twbs/bootstrap

```
Sample databases
> https://www.w3resource.com/sql/sql-table.php
> https://www.w3resource.com/sql/sample-database-of-sql-in-mysql-format.txt

S'exercer en POO
> https://phpenthusiast.com

PHPStorm
> jetbrains.com
> Pour la license gratuite, il suffit de demander une license educative dans la gestion du compte

## Programmation orientée objet

### Rappels

#### Déclarer une variable
```php
<?php
$nomDeLaVariable = "Valeur";
```

Utiliser une variable
```

```php
<?php
// On a juste à appeller notre variable avec son nom
echo $variable;
```

Fonction

```php
<?php
function nomDeLaFonction ($paramètre) {
    echo $paramètre;
}

nomDeLaFonction('Test');
//Affichage de Test
```

Condition

```php
<?php
if ($a > 10) {
    echo 'Supérieur à 10.';
}
else if ($a > 0) {
    echo 'Compris entre 0 et 10';
}
else {
    echo 'Inférieur à 0';
}

switch ($a) {
    case 10:
        echo 'Dix';
        break;

    case 9:
        echo 'Neuf';
        break;

    default:
        echo "Ni neuf, ni dix";
        break;
}
```

Opérateurs de comparaison

```php
<?php
// ==
// !=
// >
// <
// >=
// <=
```

Boucles

For

```php
<?php
for ($i=1; $i < $a; $i++) {
    echo $i . '<br>';
}
```

Do & While

```php
<?php
while ($a <= 10) {
    # code...
}

do {
    # code...
} while ($a <= 10);
```

Foreach

```php
<?php
$tab = ['a' => 1, 'b' => 2, 3, 4];

foreach ($tab as $key => $value) {
    echo $key . ' => ' . $value . '<br>';
}
```

Afficher toutes les erreurs php

```php
<?php
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);
```

Programmation Orientée Objet

Classe

```php
<?php

class Client {
    public $nom;
    public $prenom;

    function __construct($nom, $prenom)
    {
        $this->nom = $nom;
        $this->prenom = $prenom;
    }

    static function societe () {
        echo "Ma société DAWAN est super !";
    }
}
```

Instancier et utiliser un objet

```php
<?php
//Il faut penser à inclure le fichier contenant la classe avtn de l'appeller
include 'Classes/Client.php';
$client1 = new Client('HENRI', 'Thierry');
echo $client1->nom;
$client1->société;
```

> Il est possible d'utiliser et d'accéder à des attributs / méthodes sans instancier un objet mais il est nécessaire que ces attributs/méthodes soient déclaré(e)s de manière statique

```php
<?php
Client::societe();
```

Getter et Setter

```php
<?php
class Client {
    private $nom;

    function setNom ($nom) {
        $this->nom = $nom;
    }

    function getNom () {
        return $nom;
    }
}
```

Fonctions magiques

Les fonctions magiques sont des fonctions automatiquement appellées. Il est possible de les sircharger dans un objet.

Par exemple avec la fonction __get(), qui s'autoinvoque lorsque l'on essaye d'accéder à une propriété protégée ou privée.

```
<?php
class Client {
    private $nom = 'Test';

    function __get ($name) {
        return $this->$name;
    }
}

$client = new Client();
echo $client->nom;
//Affichera 'Test'
```

Interfaces

Une interface sert à obliger une classe qui l'implémente à définir les fonctions qu'elle contient avec une portée et un nommage précis.

```
<?php
interface Vehicule {
    const TVA = '20%';
    function accelerer();
    function getVitesse();
}

class Moto implements Vehicule {
    function accelerer() {

    }

    function getVitesse() {

    }
}

echo $moto->TVA;
echo Moto::TVA;
echo Vehicule::TVA;
```

Il est impossible de redéfinir la fonction d'une interface dans un enfant de l'objet qui l'implémente;

Traits

Un trait est composé d'une ou plusieurs méthodes personnalisées, on peut ajouter ces méthodes dans une classe en la faisant "hériter" de ce trait.

```php
<?php
Trait Turbo {
    static function turbo () {
        echo 'Je met le turbo !';
    }
}

class Moto {
    use Turbo;
}

Moto::turbo();
```

PDO & Base de donnée

L'objet PDO

PDO est une classe préconfigurée de PHP, celle-ci est utilisée pour réaliser des connexions et requêtes à une base de donnée.

Pour l'hydrater (instancier) il faut lui passer une chaîne de connexion, ainsi que les identifiants de connexion à la base de donnée.

```php
$pdo = new PDO ("mysql:host=localhost;dbname=wordpress", "user", "password");
```

Une fois instancié, l'objet PDO nous permet de requêter la bdd au travers de multiple méthodes, nous utiliserons ici `prepare()`

```php
$query = $pdo->prepare("SELECT * FROM users;");
$query->execute();
$results = $query->fetchAll();
//On utilise fetchAll() quand on récupère plusieurs
//résultats et fetch() lorsque l'on en attend qu'un.
```

Polymorphisme et Généricité

Généricité : Lorsque des Classes qui sont utilisées dans un même environnement sont libres de communiquer entre elles. Lors d'un héritage d'une classe à une autre, l'enfant peut ou non réutiliser ou remplacer les méthodes et attributs de la classe parente.

Polymorphisme : Lorsque des Classes qui sont utilisées dans un même environnement sont contrainte à le faire, et sont contraintes à le faire d'une certaine manière.

Erreurs et Exceptions

Lever une Exception

```php
throw new Exception("Message", "Code d'erreur");
```

Try Catch

Cette fonction sert à "attraper" les erreurs pendant l'execution d'une portion de code donnée.

```
try {
    $query->execute();
}
catch (Throwable $e) {
// Ici on peut récupérer des types d'exceptions plus précis
// si on le désire (ex : PDOException, Exception etc...) en
// changeant le type de l'argument
    echo $e->getMessage();
}
```

Composer

Composer est le gestionnaire de paquets php, il permet 3 choses : - Installer des projets entier grâce à leur repository - Ajouter et installer des dépendances à un projet - Gérer et Mettre à jour toutes les dépendances d'un projet

Exemple d'installation de projet ```shell= composer create-project laravel/laravel {directory} 4.2 --prefer-dist

```
Exemple d'ajout de dépendance / package
```shell=
composer require twbs/bootstrap
```

Exemple de mise à jour de toutes les dépendances ```shell= composer update
```

```
> L'un des avantage de composer concerne le versionning (git etc.), en effet cela permet de ne pas ver

### Authentification & Contrôle d'accès

Le principe :
Nous allons utiliser la session ($_SESSION) pour stocker les infos de connexion de l'utilisateur.

Quand l'utilisateur se connectera, on stockera une information dans la session.

Sur chaque page on vérifiera que l'information est présente dans la session, si oui, alors la personne

### Exercices

#### 1

Créer une fonction somme qui prend 2 paramètres de type int et retourne la somme des deux paramètres.

#### 2

Créer une fonction qui prend un tableau en paramètre, et qui retourne le plus grand élément du tableau

#### 3

Créer une fonction qui renvoi le nom de la Capitale des pays suivants :
France, Allemagne, Italie, Espagne, Portugal.

#### 4

Créer une classe de sorte à ce que le code suivant fonctionne :
```php
$vehicule = new Voiture('Renault', 'Laguna', 120);
$vehicule->presentation();

//la fonction doit afficher : Cette voiture est une Renault Laguna de 120 chevaux.

Voiture::commander();

//la fonction doit afficher : Pour commander un uber, rendez-vous sur uber.com
```

5

Tous les attributs utilisés doivent être définis comme privé.

Créer 3 classes - Mage - Guerrier - Voleur

Chacune des classes doit avoir les attributs suivants - Nom - Points de vie - Points d'attaque

Chacune de ses classes doit avoir une fonction `attaque($cible)` qui retire x points de vie à la cible, x étant la valeur de l'attribut attaque de l'objet en question et affiche ('Nom de l'objet 1' attaque 'Nom de l'objet 2' avec x Points de dégats)

Bonus : si les points de vie d'un objet sont nul ou négatifs après l'attaque d'un autre objet, celui-ci doit être détruit et afficher une
```

phrase du type : 'nom de l'objet' est mort.

6

Considérons les classes suivantes : - `Vehicule` qui a pour attributs : - `$nom` protégé de type string - `$acceleration` protégé de type int - `$freinage` protégé de type int - `$marque` protégé de type string - `$vitesse` protégé de type int qui a pour valeur de base 0 - `VehiculeVolant` qui étend Vehicule - `Moto` qui étend Vehicule - `Bateau` qui étend Vehicule - `Avion` qui étend VehiculeVolant - `Helicoptère` qui étend VehiculeVolant - `Voiture` qui étend Vehicule et qui a pour attributs : - `$nbPortes` protégé de type int

1. Dans des fichiers différents, créer les classes avec leurs constructeurs, setters et getters nécessaires (voir plus tard les attributs a modifier);
2. Ajouter une méthode `accélérer()` à `Vehicule` qui ajoute son `$acceleration` à sa `$vitesse`
3. Ajouter une méthode `wheeling()` à `Moto` qui retourne la phrase : "[NOM DU VEHICULE] lève sa roue avant !" seulement si la `$vitesse` est supérieure à 0
4. Ajouter une méthode `crash()` à `VehiculeVolant` qui retourne la phrase :"[NOM DU VEHICULE] s'est crashé(e) !"
5. Ajouter une méthode `coule()` à `Bateau` qui retourne la phrase : [NOM DU VEHICULE] à coulé(e)"
6. Surcharger la méthode `accélérer()` dans `Avion` afin que celle-ci multiplie la `$vitesse` par `$acceleration`
7. Dans index.php, inclure les autres fichiers, instancier un véhicule de chaque type et tester les différentes méthodes.

7

Créer une classe API; Créer une classe Film;

La classe Film doit avoir 3 attributs : - nom - poster - resume

La classe API doit prendre en paramètre une clé d'API lors de son hydratation.

ex : `$api = new API ('625b3e1220c0fca7c7ac7f6fcca786ac')`

Elle doit avoir une méthode `getFilm($id)` qui renvoi un objet Film complété avec les infos récupérés sur l'API (tester avec l'id 76341 : Mad Max Fury Road)

Ajouter un champ texte et un champ submit pour pouvoir "rechercher" des films par leur titre (déjà fait dans le répo github)

Faire en sorte que la liste des films s'affiche lorsque l'on valide le formulaire.

Ajouter Bootstrap à votre projet et appliquez le par défaut sur toutes vos pages.

Faire apparaitre un titre de site en haut de la page (ex : TheMovieDb)

Afficher la liste de film sous forme de blocs contenants le poster, le titre, et le résumé.

Les films n'ayant pas de poster doivent afficher une image par défaut.

8

Créer une page qui affiche la liste des Clients

1. Créer une classe Client qui possède chaque colonne de la bdd en attribut.
2. Créer la connection à la base de donnée (par exemple créer une classe BDD qui aura des fonctions comme getAllCustomers() et qui renverra les données en provenance de la bdd, en convertissant les données brutes en objets Client)
3. Récupérer tous les clients, itérer sur ceux-ci et afficher sous forme de liste (idéalement, créer un front controller comme le Renderer)

```
shell= index.php |__Classes |__BDD.php |__Renderer.php |__Client.php |__Autoload.php
```

Créer une page formulaire pour créer des client

1. Créer la page avec le formulaire, par exemple insert-customer.php
2. Finaliser le template html customer-insert.html
3. Créer la méthode insertClient(Client $client) dans la classe BDD

4. Relier le formulaire à la méthode avec une vérification des variables $_GET ou $_POST en fonction de ce que vous avez utilisé sur la page php correspondante (insert-customer.php).

Aller plus loin

Faire un CRUD (Create Read Update Delete) complet sur les clients. Ajouter une page "Détails du client" dans laquelle on retrouve aussi les détails de son agent. Styliser les pages avec du CSS et/ou Bootstrap ou un autre framework front.

Pour le delete :

Dans index.php, ajouter une vérification : Si la variable $_GET['delete'] existe, on vérifie si le client avec le code correspondant existe, si c'est le cas, on le supprime de la base de donnée.

Pour le edit :

Refaire un nouveau template (basé sur le insert) Créer la méthode dans le rendere qui permet de passer tous les paramètre au formulaire (avec le str_replace)

Ajouter la fonction de sauvegarde soit dans $bdd, soit dans l'objet Client directement.