Programmez en orienté objet en PHP

Introduction à la Prog Orienté Objet (POO) en PHP

1. Pourquoi la POO ?
    - Prob avec le code procédural : Ajoutez du code en augmentant sans fin, qui peut mener à un code difficile à maintenir
    - Solution : La POO aide à structurer le code de manière organisé et évolutive, comme la construction d'un pont solide plutôt que de simplement empiler les briques

2. Choix de la structure :
    - Comme pour un pont, il est crucial de bien choisir la structure en POO. trop petit, et vous devez tous refaire. Trop grand, et vous pourriez gaspiller des ressources ou avoir des probs de maintenance.

3. En résumé :
    - **Avantages de la POO** : Structure le code de manière organisée et évolutive, facilitant la maintenance et l'extension des projets.
    - Importance d'une **structure appropriée** : Choisir la bonne structure est crucial pour la pérennité et la facilité d'entretien du code.
    - La POO comme solution aux limitations du code procédural, offrant une meilleure organisation et évolutivité.
    - **Concepts fondamentaux** : Objets (instances de classes) et Classes (modèles pour créer des objets), regroupant propriétés et méthodes.
    - L'instanciation comme processus de création d'objets à partir de classes définies.

Découverte des Objets et Classe en PHP

1. Objet et Classes :

- Objet = Instance d'une classe. Regroupe des propriétés (données) et des méthodes (fonctions)
- Classes = Modèle pour créer des objets, Définit les propriétés et les méthodes dispo.

2. Instanciation :
   - Créer un objet : Utilise le mot-clé "new" pour créer une instance d'une classe.

     Exemple :

     ```
     $date = new DateTime();
     ```

   - Instance : Un objet créé à partir d'une classe. Chaque instance est un objet même si elles sont issues de la même classe.

3. Référence et Modif :
   - Référence : les Objets PHP sont passés par référence, ce qui signifie que toutes les modifs de l'objet affecte l'original.

     Exemple :

     ```
     $date->modify('+1 day'); //Ici on modifie l'objet Date
     ```

   - Passage par référence : pour modifier une variable dans une fonction, utilisez le symbole '&'

     exemple :

     ```
     function foo(&$var) {
         $var = 2;
     }
     ```

   - Comparaison : Utilisez ' ==' pour comparer les valeurs et '===' pour comparer les instances (si elles sont le même objet en mémoire)

4. Accès aux Propriétés et Méthode $

- Accès : Utilisez ‘ - >’ pour accéder aux propriétés et méthodes d’un objet.

  Exemple :

```
$date->format('d/m/Y');
```

- Chaînage (Chaining) : Appel plusieurs méthodes successivement sur le même objet

  Exemple :

```
$formatedDate = $date->modify('+1 day')->format('d/m/Y'
```

5. stdClass et JSON

- stadClass : Classe générique utilisé par PHP pour les objets JSON non typé.

  Exemple :

```
$s = '{"date":"2021-03-23","timezone":"Europe/Paris"}';
$obj = json_decode($s);
```

- JSON Décodé : PHP utilise stdClass lorsque la classe spécifique n’est pas définie.

6. Méthode Retour :

- Retour d’Objet : Les méthodes peuvent retourner l’instance actuelle ou une nouvelle instance, en facilitant le chainage.

  Exemple :

```
$newDate = $date->modify('+1 day'); // $newDate est l'o
```

Résumé :

- **Classe :** Modèle avec propriétés et méthodes.
- **Objet :** Instance d'une classe, créée avec `new`.

- **Instance :** Objet spécifique créé à partir d'une classe.
- **Accès et Manipulation :** Utilisez `>` pour accéder aux propriétés et méthodes. Les objets sont passés par référence.
- **Chaînage :** Appeler plusieurs méthodes sur le même objet en série.
- **stdClass :** Utilisée pour les objets JSON sans classe définie.