Développement PHP
Partie 7 : Programmation Orientée Objet

DENIS LOEUILLET – IFA - 2017

Partie 7 : Programmation Orientée Objet
Les bases de la POO

1. Introduction à la POO
2. Utiliser la classe
3. L'opérateur de résolution de portée
4. Manipulation de données stockées
5. L'héritage
6. Les méthodes magiques

Partie 7 : Programmation Orientée Objet
Les bases de la POO

3. L'opérateur de résolution de portée

- Symbole : « :: »
- Nom : « double deux points » (« Scope Resolution Operator » en anglais)
- Utilisation : pour appeler des éléments appartenant à telle classe et non à tel objet.
  - nous pouvons définir des attributs et méthodes appartenant à la classe :
    - ce sont des éléments statiques.
- Parmi les éléments appartenant à la classe (et donc appelés via cet opérateur) :
  - il y a aussi les « constantes de classe », sortes d'attributs dont la valeur est constante,
  - c'est-à-dire qu'elle ne change pas.
- Cet opérateur est aussi appelé « Paamayim Nekudotayim ».

Partie 7 : Programmation Orientée Objet
Les bases de la POO : opérateur de résolution de portée

- Les constantes de classe
  - Le principe est à peu près le même que lorsque vous créez une constante à l'aide de la fonction define.
  - Les constantes de classe permettent d'éviter tout code muet. Voici un code muet :

```php
<?php
$perso = new Personnage(50);
```

    - ✓ Pourquoi est-il muet ?
      - ❖ parce qu'on ne sait pas à quoi « 50 » correspond.
    - ✓ Qu'est-ce que cela veut dire ?
      - ❖ Étant donné que je viens de réaliser le script, je sais que ce « 50 » correspond à la force du personnage. Cependant, ce paramètre ne peut prendre que 3 valeurs possibles :
        - 20, qui veut dire que le personnage aura une faible force ;
        - 50, qui veut dire que le personnage aura une force moyenne ;
        - 80, qui veut dire que le personnage sera très fort.

Partie 7 : Programmation Orientée Objet
Les bases de la POO : opérateur de résolution de portée

- Les constantes de classe
  - Le principe est à peu près le même que lorsque vous créez une constante à l'aide de la fonction define.
  - Les constantes de classe permettent d'éviter tout code muet. Voici un code muet :
    - ✓ Au lieu de passer ces valeurs telles quelles, on va plutôt passer une constante au constructeur.
      - ❖ Ainsi, quand on lira le code, on devinera facilement que l'on passe une force moyenne au constructeur.
      - ❖ C'est plus facile à comprendre qu'un nombre quelconque.
  - Une constante :
    - ✓ est une sorte d'attribut appartenant à la classe dont la valeur ne change jamais.
  - Pour déclarer une constante, vous devez faire précéder son nom du mot-clé const.
    Script : declaration-constante.php
  - Quel rapport avec les « double deux points » ?
    - ✓ Contrairement aux attributs :
      - ❖ vous ne pouvez accéder à ces valeurs via l'opérateur « -> » depuis un objet (ni $this ni $perso ne fonctionneront)
      - ❖ mais avec l'opérateur « :: » car une constante appartient à la classe et non à un quelconque objet.

Partie 7 : Programmation Orientée Objet
Les bases de la POO : opérateur de résolution de portée

- Les constantes de classe
  - Pour accéder à une constante, vous devez :
    - ✓ spécifier le nom de la classe,
    - ✓ suivi du symbole double deux points,
    - ✓ suivi du nom de la constante.
    - ✓ Exemple : script acces-constante.php
  - Notez qu'ici les noms de constantes sont en majuscules :
    - ✓ c'est encore et toujours une convention de dénomination.

Partie 7 : Programmation Orientée Objet
Les bases de la POO : opérateur de résolution de portée

- Les attributs et méthodes statiques : les méthodes statiques
  - Les méthodes statiques sont des méthodes qui sont faites pour agir sur une classe et non sur un objet.
    - ✓ Par conséquent, on ne peut pas utiliser $this dans la méthode !
    - ✓ En effet, la méthode n'étant appelée sur aucun objet, il serait illogique que cette variable existe.
    - ✓ Souvenez-vous : $this est un paramètre implicite. Cela n'est vrai que pour les méthodes appelées sur un objet !
  - Remarque :
    - ✓ Même si la méthode est dite « statique » :
      - ❖ il est possible de l'appeler depuis un objet : $obj->methodeStatique()
      - ❖ mais, même dans ce contexte, la variable $this ne sera toujours pas passée !
  - Pour déclarer une méthode statique, vous devez faire précéder le mot-clé function du mot-clé static après le type de visibilité.
  - Cependant, préférez appeler la méthode avec l'opérateur « :: » comme le montre le premier de ces deux codes.
    - ✓ De cette façon, on voit directement de quelle classe on décide d'invoquer la méthode.

Partie 7 : Programmation Orientée Objet
Les bases de la POO : opérateur de résolution de portée

- Les attributs et méthodes statiques : les attributs statiques
  - Le principe est le même, c'est-à-dire qu'un attribut statique appartient à la classe et non à un objet.
  - Ainsi, tous les objets auront accès à cet attribut et cet attribut aura la même valeur pour tous les objets.
  - La déclaration d'un attribut statique se fait en faisant précéder son nom du mot-clé static.
    Script attribut-statique.php

Partie 7 : Programmation Orientée Objet
Les bases de la POO : opérateur de résolution de portée

- Les attributs et méthodes statiques : les attributs statiques
  - à quoi sert un attribut statique ?
    - ✓ Nous avons vu que les méthodes statiques sont faites pour agir sur la classe.
      - ✓ D'accord, mais qu'est-ce qu'on peut faire sur une classe ?
        - ❖ modifier les attributs de celle-ci car des attributs appartenant à une classe ne sont autre que des attributs statiques !
        - ❖ Les attributs statiques servent en particulier à avoir des attributs indépendants de tout objet.
          - Ainsi, votre attribut aura toujours la même valeur (sauf si l'objet modifie sa valeur, bien sûr).
          - Mieux encore : si l'un des objets modifie sa valeur, tous les autres objets qui accéderont à cet attribut obtiendront la nouvelle valeur !
          - C'est logique quand on y pense, car un attribut statique appartenant à la classe, il n'existe qu'en un seul exemplaire.
            - Si on le modifie, tout le monde pourra accéder à sa nouvelle valeur.

Partie 7 : Programmation Orientée Objet
Les bases de la POO : opérateur de résolution de portée

- Les attributs et méthodes statiques : les attributs statiques
  - La ligne 51 commence avec le mot-clé self, ce qui veut dire (en gros) « moi-même » (= la classe).
  - Notre ligne veut donc dire :
    - ✓ « Dans moi-même, donne-moi l'attribut statique $_texteADire. »
  - Remarque :
    - ✓ N'oubliez pas de mettre un « $ » devant le nom de l'attribut.
    - ✓ C'est souvent source d'erreur donc faites bien attention.
  - Exercice : script compter-instances.php
    - ✓ Créez une classe toute bête qui ne sert à rien.
    - ✓ Seulement, à la fin du script, je veux pouvoir afficher le nombre de fois où la classe a été instanciée.
    - ✓ Pour cela, vous aurez besoin d'un attribut appartenant à la classe (admettons $_compteur) qui est incrémenté dans le constructeur.

Partie 7 : Programmation Orientée Objet
Les bases de la POO : opérateur de résolution de portée

- Les attributs et méthodes statiques : les attributs statiques
  - Exercice : script compter-instances.php
    - ✓ Eh oui, le retour du mot-clé self…
    - ✓ Pourquoi pas $this ?
      - ❖ Souvenez-vous : on n'accède pas à un attribut statique avec $this mais avec self !
        - self représente la classe tandis que $this représente l'objet actuellement créé.
        - Si un attribut statique est modifié, il n'est pas modifié uniquement dans l'objet créé mais dans la structure complète de la classe !
        - Je me répète, mais il est essentiel que vous compreniez ça, c'est très important.

Partie 7 : Programmation Orientée Objet
Les bases de la POO : opérateur de résolution de portée

- Résumé :
  - L'opérateur « -> » permet d'accéder à un élément de tel objet, tandis que l'opérateur « :: » permet d'accéder à un élément de telle classe.
  - Au sein d'une méthode, on accède à l'objet grâce à la pseudo-variable $this, tandis qu'on accède à la classe grâce au mot-clé self.
  - Les attributs et méthodes statiques ainsi que les constantes de classe sont des éléments propres à la classe, c'est-à-dire qu'il n'est pas utile de créer un objet pour s'en servir.
  - Les constantes de classe sont utiles pour éviter d'avoir un code muet, c'est-à-dire un code qui, sans commentaire, ne nous informe pas vraiment sur son fonctionnement.
  - Les attributs et méthodes statiques sont utiles lorsque l'on ne veut pas avoir besoin d'un objet pour s'en servir.