Constantes de classe, méthodes et attributs statiques

Programmation Orientée Object (POO) en PHP

Chris Chevalier – chevalier@chris-freelance.com

L'opérateur de résolution de portée

Constantes de classe, méthodes et attributs statiques

L'opérateur de résolution de portée en quelques mots…

- L'**opérateur de résolution de portée** (ou **Scope Résolution Operator**) est représenté par un **double deux points** : « :: »
- Contrairement à l'**opérateur flèche** (« -> »), il est utilisé pour appeler des **attributs** et **méthodes** appartenant à la **classe** et non à l'**objet**
- Il est effectivement possible de définir des éléments propres à une **classe PHP** - Il s'agit d'éléments **statiques**
- Parmi ces éléments, on retrouve les **constantes de classe** représentant des attributs à la valeur… **constante**

Les constantes de classe

Constantes de classe, méthodes et attributs statiques

Les constantes de classe en quelques mots… (1/2)

- Comme les **constantes PHP** déclarées à l'aide de la **fonction** *define()*, les **constantes de classe** permettent de définir des valeurs qui ne seront pas amenées à changer
- En l'occurence, elle vont surtout nous servir à éviter d'écrire du **code muet**
- Voici un **code muet** :

```
$gandalf = new Personnage("Gandalf", 120);
```

- Ce code est muet parce qu'on ne sait pas à quoi correspondant la valeur *120*

Les constantes de classe en quelques mots… (2/2)

- Plutôt que de passer ce genre de valeur, il peut être plus intéressant de passer une **constante** au **constructeur**
- Un nom de **constante** sera toujours plus évocateur qu'une simple valeur à la lecture du code
- En pratique la déclaration d'une **constante de classe** se fait très simplement puisqu'il suffit de placer le mot-clé *const* devant son nom
- Notez également que, contrairement aux **variables PHP**, une **constante** ne prend pas de $ devant son nom

Les constantes de classe en image...

```php
class Personnage {
    // Attributs
    private $nom;
    private $pointsDeVie;
    private $pointsDeMagie;

    const PV_FAIBLE = 50;
    const PV_MOYEN = 80;
    const PV_ELEVE = 120;

    // Méthodes
    public function __construct() {

    }
}
```

Accéder à une constante de classe (1/2)

- Remarquez que, **par convention**, on écrit toujours le nom de nos **constantes** en majuscule
- Si nous souhaitons maintenant accéder à la valeur stockée dans une de ces **constantes**, on pourra pas utiliser l'**opérateur** -> mais bien ::
- Retenez qu'une **constante** appartient à la **classe** et non à un **objet**
- Plus concrètement, on y accède en spécifiant le nom de la **classe** suivi du symbole de **double deux points** puis de son nom (*MaClasse::MACONSTANTE*)

```php
class Personnage {
  // Attributs
  private $nom;
  private $pointsDeVie;
  private $pointsDeMagie;

  const PV_FAIBLE = 50;
  const PV_MOYEN = 80;
  const PV_ELEVE = 120;

  // Méthodes
  public function __construct($pv) {
    $this->setPointsDeVie($pv);
  }

  public function setPointsDeVie($pv)
  {
    if (in_array($pv, [self::PV_FAIBLE, self::PV_MOYEN, self::PV_ELEVE]))
    {
      $this->pointsDeVie = $pv;
    }
  }
}
```

Explications

- On initialise maintenant le nombre de points de vie du *Personnage* à sa **construction**
- Pour ce faire, on fait appel à la **méthode** (**setter**) *setPointsDeVie()* depuis le **constructeur** afin de mettre à jour la valeur de l'**attribut privé** *$pointsDeVie*
- On s'assure, dans le **setter**, que la valeur passée en **argument** correspond bien à une des 3 **constantes de classe** déclarées plus haut
- Remarquez que l'on utilise pas le mot-clé *$this* pour accéder à une **constante** mais *self* qui représente la **classe** elle-même lorsque l'on se situe à l'intérieur
- Si c'est le cas, on peut effectivement mettre à jour la valeur

Accéder à une constante de classe (2/2)

- Dans la même idée, on peut accéder simplement à une **constante de classe** depuis l'extérieur de celle-ci

- Créons une nouvelle **instance** de la **classe** *Personnage*

```php
$perso = new Personnage(Personnage::PV_ELEVE);
```

Les attributs et méthodes statiques

Constantes de classe, méthodes et attributs statiques

Les méthodes statiques

- Comme les **constantes**, une **méthode statique** est un élément propre à la **classe** et non à un **objet** issu de celle-ci

- Par conséquent, une **méthode statique** ne peut pas utiliser le mot-clé *$this* puisque, pour rappel, *$this* représente l'**objet** depuis lequel est appelée une **méthode** - Or, on appellera nos **méthodes statiques** depuis une **classe** et non un **objet**

- Pour déclarer une **méthode statique**, il suffit de placer le mot-clé *static* entre le **type de visibilité** et le mot-clé *function* (*public* ***static*** *function maFonction() { }*)

```php
class Personnage {
  // Attributs
  private $nom;
  private $pointsDeVie;
  private $pointsDeMagie;

  const PV_FAIBLE = 50;
  const PV_MOYEN = 80;
  const PV_ELEVE = 120;

  // Méthodes
  public function __construct($pv) {
    $this->setPointsDeVie($pv);
  }

  public function setPointsDeVie($pv)
  {
    if (in_array($pv, [self::PV_FAIBLE, self::PV_MOYEN, self::PV_ELEVE]))
    {
      $this->pointsDeVie = $pv;
    }
  }

  public static function attaquer() {
    echo "Le personnage attaque";
  }
}
```

Accéder à une méthode statique

- Notez enfin que, malgré son appartenance à la **classe** et non à un **objet**, une **méthode statique** peut être appelée depuis un **objet** issu de sa **classe**

- Ainsi, ces 2 appels fonctionnent et produisent le même résultat :

```
31    $perso = new Personnage(Personnage::PV_ELEVE);
32
33    Personnage::attaquer();
34    $perso->attaquer();
```

- Je vous invite, néanmoins, à favoriser la première approche afin d'expliciter la **classe** depuis laquelle est appelée la **méthode** et ainsi éviter toute erreur

- Malgré cet aspect, le mot-clé *$this* reste inaccessible au sein d'une **méthode statique** étant donnée que, par définition, elle agit uniquement sur la **classe** et non sur un **objet** quel qu'il soit

Les attributs statiques

- Le principe est le même pour les **attributs statiques**
- Un **attribut statique** appartient donc à la **classe** dans laquelle il est déclaré et non à un **objet**
- De cette manière, tous les **objets** issus de cette **classe** ont accès à cet **attribut** partageant une valeur commune à tous ces **objets**
- Comme une **méthode statique**, un **attribut statique** est déclaré à l'aide du mot-clé *static* situé entre le **type de visibilité** et le nom de la variable (*public **static** $monAttribut*)

```php
class Personnage {
  // Attributs
  private $nom;
  private $pointsDeVie;
  private $pointsDeMagie;

  const PV_FAIBLE = 50;
  const PV_MOYEN = 80;
  const PV_ELEVE = 120;

  private static $attaque = "Le personnage attaque";

  // Méthodes
  public function __construct($pv) {
    $this->setPointsDeVie($pv);
  }

  public function setPointsDeVie($pv)
  {
    if (in_array($pv, [self::PV_FAIBLE, self::PV_MOYEN, self::PV_ELEVE]))
    {
      $this->pointsDeVie = $pv;
    }
  }

  public static function attaquer() {
    echo self::$attaque;
  }
}
```

Explications

- L'intérêt des **attributs privés** réside dans le fait qu'ils conservent leur valeur d'un **objet** à l'autre puisqu'elle correspond à la **classe** et non à l'**objet** - Ils sont dont indépendants de tout **objet**

- Évidemment, il s'agit d'une variable - Ainsi, si l'on choisit de modifier sa valeur, tous les **objets** auront accès à cette nouvelle valeur

- Ainsi, la ligne 29 de l'exemple précédent utilise le mot-clé *self* et l'**opérateur de résolution de portée** pour accéder à la valeur stockée dans l'**attribut statique** *$attaque*

- N'oubliez pas de toujours placer un *$* devant le nom de l'**attribut statique** que vous appeler sous peine de rencontrer certaines erreurs

TP - Compteur d'instances

Constantes de classe, méthodes et attributs statiques

TP - Compteur d'instances

- Déclarez une **classe** *Count* contenant un **attribut statique** *$count* incrémenté à chaque **instanciation** de cette **classe**

- Prévoyez un **getter statique** de l'**attribut** *$count* afin d'afficher le résultat obtenu après plusieurs **instanciations** dans votre navigateur à l'aide de la **fonction PHP** *echo()* depuis le programme principal (à l'extérieur de la **classe**)

TP - Compteur d'instances

- Envoyez votre fichier *Count.php* à l'adresse email chevalier@chris-freelance.com

Des questions ?