Programmation orientée objet en PHP

Accesseurs et mutateurs

En programmation objet il existe un grand principe qui s'appelle l'encapsulation. L'encapsulation signifie que l'état d'un objet ne peut être modifié que par les méthodes de cet objet. Par "état" il faut entendre la valeur des attributs à un moment t. Prenons un exemple avec notre classe Humain : nous allons créer une instance de la classe Humain et modifier son age en accédant directement à l'attribut age via l'opérateur ->. Ensuite nous afficherons la valeur de l'attribut age :

```php
<?php

// Humain.php
class Humain {

    public $age;
    public $taille;
    public $poids;

    public function __construct($page, $ptaille, $ppoids) {
        $this->age = $page;
        $this->taille = $ptaille;
        $this->poids = $ppoids;
    }

}

$sparrow = new Humain(42, 1.84, 75);
$sparrow->age = 456;
echo "age de Sparrow = $sparrow->age";
```

Ici nous avons pu modifier directement la valeur de l'attribut sans passer par une méthode de Humain. Cela est en contradiction avec le principe d'encapsulation. Si nous voulons rendre impossible l'accès à un attribut ou une méthode en dehors de la classe, nous devons utiliser le mot clé "private", ou lieu du mot clé "public", devant les attributs et méthodes concernés. Ici nous souhaitons simplement "protéger" les trois attributs, notre classe devient donc :

```php
<?php

// Humain.php
class Humain {

    private $age;
    private $taille;
    private $poids;

    public function __construct($page, $ptaille, $ppoids) {
        $this->age = $page;
        $this->taille = $ptaille;
        $this->poids = $ppoids;
    }

}

$sparrow = new Humain(42, 1.84, 75);
$sparrow->age = 456; // déclenche une erreur !
```

```
20 echo "age de Sparrow = $sparrow->age";
```

Les mots clés public et private sont des modificateurs d'accès, ils définissent le niveau d'accessibilité ou de visibilité d'un attribut ou d'une méthode. Le mot clé public signifie que l'attribut ou la méthode est accessible partout où la classe est accessible. Le mot clé private signifie que l'attribut ou la méthode n'est accessible qu'au sein de la classe. Il en existe un troisième : protected, dont nous reparlerons bientôt.

Mais comment allons-nous faire maintenant si nous souhaitons récupérer ou modifier un attribut dans notre programme ? Nous allons créer des méthodes spécifiques : une méthode pour récupérer la valeur d'un attribut : un getter ou accesseur, une méthode pour assigner une valeur à un attribut : un setter ou mutateur. Chaque attribut pourra donc posséder un getter et un setter, ou l'un des deux, ou aucun, c'est selon...
En POO l'usage est de nommer les getters get*Attribut* (par exemple getNom) et les setters set*Attribut* (par exemple setAge). Pour les booléens, le getter s'appellera is*Attribut* (par exemple isValid).
Le getter est donc une simple méthode sans paramètre qui va renvoyer la valeur d'un attribut. Le setter est une méthode qui assignera à un attribut une nouvelle valeur passée en argument. La classe Humain devient donc :

```php
<?php

// Humain.php
class Humain {

    private $age;
    private $taille;
    private $poids;

    public function __construct($page, $ptaille, $ppoids) {
        $this->age = $page;
        $this->taille = $ptaille;
        $this->poids = $ppoids;
    }

    public function getAge() {
        return $this->age;
    }

    public function setAge($age) {
        $this->age = $age;
    }

}

$sparrow = new Humain(42, 1.84, 75);
$sparrow->setAge(456);
echo "age de Sparrow = {$sparrow->getAge()}";
```

Et maintenant nous pouvons modifier et récupérer la valeur de l'attribut en utilisant le getter et le setter.

PHP permet de définir un getter et un setter unique pour tous les attributs. Le getter sera appelé automatiquement lorsque l'on tentera d'accéder directement à un attribut inaccessible (avec l'opérateur ->), le setter sera appelé automatiquement lorsque l'on tentera de modifier directement la valeur d'un attribut inaccessible (avec l'opérateur ->).
Le getter prendra en argument le nom de l'attribut dont on souhaite récupérer la valeur. Le setter prendra en argument l'attribut dont on souhaite modifier la valeur et la nouvelle valeur à assigner.

```php
<?php

// Humain.php
class Humain {

    private $age;
    private $taille;
    private $poids;

```

```php
    public function __construct($page, $ptaille, $ppoids) {
        $this->age = $page;
        $this->taille = $ptaille;
        $this->poids = $ppoids;
    }

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

    public function __set($name, $value) {
        $this->$name = $value;
    }

}

$sparrow = new Humain(42, 1.84, 75);
$sparrow->age = 456;
echo "age de Sparrow = $sparrow->age" . PHP_EOL;
$sparrow->taille = 2;
echo "la taille de Sparrow est $sparrow->taille";
```

Fin

Les tests avec PHPUnit
Petite application version 2
Petitie application version 3
Mes cours

ADMINISTRATION

Administration du cours
Réglages de mon profil

Connecté sous le nom « Arnaud Lemais » (Déconnexion)
POO PHP