PHP Objet
INTRODUCTION

Juin 2018

1.

Définitions

Qu’est-ce que la programmation objet ?

Qu’est-ce qu’une classe ?

Une classe est un ensemble de propriétés et de méthodes

Objet DateTime

- Formater une date
- Calculer un interval
- Comparer deux dates

Objet Mail

- Définir les destinataires
- Créer le corp du mail
- Envoyer

Objet Image

- Redimensionner
- Appliquer un filtre
- Obtenir la largeur

Vocabulaire

Une **classe** est le code contenant la définition des méthodes et propriétés

Un **objet** est une instance de classe

La pseudo variable **$this** fait référence à l'objet

Créer un objet

```php
class Article
{
    public $date;
    public $title;
    public $content;

    public function printTitle() {
        echo $this->title;
    }
}
```

```php
$article = new Article;
$article->title =
    "Apprendre les objets";
$article->content =
    "Un objet est une instance de
classe";
echo $article->printTitle();
```

2.

Le constructeur

Un méthode pour l’instanciation

Le constructeur

Un constructeur est une méthode dite “magique”

Elle est appelée automatiquement lors de l’instanciation d’un objet

Constructeur

```php
public $active;

// Constructeur appelé à la création
// d'une instance (avec l'opérateur 'new')
public function __construct()
{
    // Par défaut active et à 1
    $this->active = 1;
}
```

Le constructeur

Il est possible de mettre des attributs pour initialiser les valeurs de l'objet

```php
public function __construct($title) {
    $this->title = $title;
};
```

```php
$article = new Article("Titre");
```

3.

Les propriétés et méthodes privés

Pour une utilisation dans la classe

C’est privé !

Les propriétés et méthodes privées ne peuvent pas être utilisée en dehors de la classe.

C’est privé !

```php
private $date;

public function __construct() {
    $this->date = new DateTime;
}

public function showDate() {
    echo $this->date
        ->format('d/m/Y');
}
```

```php
$article = new Article;
// echo $article->date;
// Erreur, l'attribut date est privé !
$article->showDate(); // C'est mieux !
```

4.

Les constante

Des valeurs qui ne changent pas

Les valeurs constantes

C’est valeurs ne sont pas modifiables

Il n’y a pas le symbole **$** pour déclarer une constante

On utilise le mot clé **self** suivis de l’opérateur de résolution de portée **::** pour utiliser une constante à l’intérieur de la classe

Pour l'extérieur, le nom de la classe est utilisé

Les constantes

```php
private $status;
const PUBLIC = 1;
const PRIVATE = 0;

public function isPublic() {
    return $this->status == self::PUBLIC;
}
```

```php
$art = new Article;
echo Article::PUBLIC;
echo $art->isPublic();
```

5.

Les propriétés et méthodes statiques

Pas besoin d’instance

Les propriétés et méthodes statiques

Les propriétés et méthodes statiques permettent d'y accéder sans instancier la classe

La pseudo variable **$this** n'est pas disponible dans une méthode statique

Une propriété statique est partagée entre les instances de classe

Elles sont utilisées comme une constante (avec **::**)

Les attributs et méthodes statiques

```php
private static $counter = 0;

public function __construct() {
    self::$counter++;
}

public static function getCount() {
    return self::$counter;
}
```

```php
echo Article::getCount(); // 0
$art = new Article;
echo Article::getCount(); // 1
```

6.

Les “getters” et “setters”

Définir les propriétés proprement

Les getters et les setters

Les propriétés d’une classe sont la plupart du temps en private pour pouvoir contrôler les valeurs

On utilise donc des méthodes pour contrôler les valeur avant de les enregistrer

Ce sont les “setters” et les “getters” pour récupérer ces valeurs

Getters et setters

```php
class Article
{
    private $title;

    public function setTitle($title) {
        if (strlen($title) < 3) { trigger_error('Le titre est trop court');}
        else { $this->title = $titre; }

        return $this;
    }
    public function getTitle() { return $this->title; }
}
```

7.

Typage objet

Définir le type d’un attribut

Le typage objet

Permet de forcer le type d’un attribut d’une méthode

Crée une erreur si le type d’objet spécifié n’est pas envoyé

Typage objet

```php
class Livre
{
    public function setAuteur(Auteur $auteur)
    {
        $this->auteur = $auteur;
    }
}

$livre = new Livre();
$livre->setAuteur('Marc Levy'); // Erreur
```