DÉVELOPPEMENT WEB CÔTÉ SERVEUR (PHP)

Programmation orientée objet

PROGRAMMATION ORIENTÉE OBJET (POO) EN PHP

La programmation orientée objet (POO) est une approche de programmation basée sur le concept d'objets et de classes.

En PHP, la POO est largement utilisée pour créer des applications Web et d'autres types de logiciels.

Voici les principaux concepts de la POO en PHP :

- **Classe** : Une classe est un modèle ou un plan pour créer des objets. Elle définit les propriétés (variables) et les méthodes (fonctions) que chaque objet de cette classe possédera.
- **Objet** : Lorsqu'une class est instanciée à l'aide du mot-clé 'new', l'instance créée est appelée un 'objet'.
  Un objet est une réalisation concrète d'une classe, contenant ses propres données (propriétés) et étant capable d'utiliser les méthodes définies dans cette classe.

C’EST QUOI UN CONSTRUCTEUR ?

Un constructeur est une méthode spéciale dans une classe qui est automatiquement appelée quand on crée un nouvel objet à partir de cette classe.

C’est comme une première chose qu’une classe fait pour préparer ou configurer l’objet. Il initialise les propriétés de l’objet ou effectue des actions nécessaires dès le départ.

Points importants :

1. Le constructeur s’appelle toujours __construct() en PHP.

2. Il est appelé automatiquement quand on utilise new.

EXEMPLE

En PHP, la syntaxe de la POO est relativement simple et intuitive. Voici un exemple de déclaration de classe en PHP :

La classe

```php
class Person{
    // Properties
    public $firstName;
    public $lastName;

    // Constructor
    public function __construct($firstName, $lastName){
        $this->firstName = $firstName;
        $this->lastName = $lastName;
    }

    // Functions
    public function getFulleName(){
        return $this->firstName . " " . $this->lastName;
    }
}
```

L'objet

```php
$pers1 = new Person("John", "Doe");
echo $pers1->getFulleName();
```

ENCAPSULATION

En PHP, l'encapsulation est réalisée à l'aide de modificateurs d'accès tels que ‘**public’**, ‘**protected’** et ‘**private’**, qui contrôlent l'accès aux propriétés et aux méthodes d'une classe.

```
class Animal {
    public $name; // Accès partout
    private $species; // Accès uniquement dans la classe Animal
    protected $color; // Accès dans Animal et ses classes dérivées

    public function __construct($name, $species, $color) {
        $this->name = $name;
        $this->species = $species;
        $this->color = $color;
    }

    // Méthode publique
    public function showDetails() {
        return "Name: $this->name, Species: $this->species, Color: $this->color";
    }
}
```

HÉRITAGE

Le mécanisme par lequel une classe peut hériter des propriétés et des méthodes d'une autre classe.
En PHP, l'héritage est déclaré à l'aide du mot-clé **extends.**

Exemple Simple:

```
class Cat extends Animal {
	public function getCatName(){
		return $this->name;
	}
}

$myDog = new Animal("Spike", "Dog", "Black");
echo $myDog->showDetails();
$myCat = new Cat("Heaven", "Cat", "White");
echo $myCat->showDetails();
echo $myCat->getCatName();
```

HÉRITAGE

Autre exemple

```php
class Dog extends Animal {
    public $breed;

    public function __construct($name, $species, $color, $breed) {
        parent::__construct($name, $species, $color); // Appel au constructeur parent
        $this->breed = $breed;
    }

    // Méthode publique supplémentaire
    public function bark() {
        return "$this->name barks! 🐕";
    }

    // Accès à une propriété protégée
    public function getColor() {
        return "$this->name has the color $this->color.";
    }

    public function getSpecies(){
        return "Species: $this->species"; // Error
    }
}
```

POLYMORPHISME

La capacité d'un objet à prendre différentes formes. En PHP, le polymorphisme peut être réalisé via la substitution de méthodes, l'interface ou l'implémentation de méthodes magiques comme __toString().

```
class Animal{
	public function talk(){
		echo "I'm an animal";
	}
}
class Dog{
	public function talk(){
		echo "Wof Wof!";
	}
}

$spike = new Dog();
$spike->talk();
```

ABSTRACTION

La POO en PHP permet également de créer des classes abstraites et des interfaces, qui définissent des méthodes sans les implémenter directement. Cela permet de créer des modèles génériques pour d'autres classes

```php
interface Animal{
        public function talk();
}
class Dog implements Animal{
    // C'est obligatoire de créer la fonction "parler()"
    public function talk(){
        echo "I'm an animal -> a dog";
    }

    // Ce n'est pas obligatoire de créer la fonction "jouer()"
    public function play(){
        echo "I like to play";
    }
}

$spike = new Dog();
$spike->talk(); // Affiche "I'm an animal -> a dog"
$spike->play(); // Affiche "I like to play"
```