Introduction à la Programmation Orientée Objet en PHP

Introduction à la Programmation Orientée Objet (POO)

La programmation orientée objet (POO) est un paradigme de programmation qui repose sur l'utilisation d'objets. En PHP, elle permet de modéliser des concepts du monde réel en termes de classes et d'objets, en facilitant la réutilisation et la maintenance du code.

Création de Classes et Objets

Une classe est un modèle qui définit les propriétés et méthodes des objets. Voici un exemple de classe en PHP :

```
class Activite {
    private $id;
    private $titre;
    private $description;
    private $prix;
    private $dateDebut;
    private $dateFin;
    private $placesDisponibles;
    private $type;

    public function __construct($id = null, $titre,
    $description, $prix, $dateDebut, $dateFin,
    $placesDisponibles, $type) {
        $this->id = $id;
        $this->titre = $titre;
        $this->description = $description;
        $this->prix = $prix;
```

```
        $this->dateDebut = $dateDebut;
        $this->dateFin = $dateFin;
        $this->placesDisponibles = $placesDisponibles;
        $this->type = $type;
    }
}
```

Listing 1: Exemple de classe Activite

Pour créer un objet à partir d'une classe :

```
$activite = new Activite(null, "Randonn e", "Une belle
    randonn e", 100, "2024-01-01", "2024-01-02", 20, "Sport")
    ;
```

Encapsulation et Modificateurs d'Accès

L'encapsulation consiste à restreindre l'accès direct aux propriétés d'une classe et à fournir des méthodes pour interagir avec celles-ci. En PHP, cela est réalisé à l'aide des modificateurs d'accès :

- **public** : accessible depuis n'importe où.
- **protected** : accessible uniquement depuis la classe elle-même et ses sous-classes.
- **private** : accessible uniquement depuis la classe elle-même.

Exemple :

```
class Exemple {
    private $valeur;

    public function setValeur($valeur) {
        $this->valeur = $valeur;
    }

    public function getValeur() {
        return $this->valeur;
    }
}
```

Listing 2: Utilisation des modificateurs d'accès

Héritage et Polymorphisme

Héritage

L'héritage permet à une classe de dériver d'une autre classe. La classe enfant hérite des propriétés et méthodes de la classe parent. En PHP, cela se fait avec le mot-clé `extends`.

```php
class User {
    protected $nom;
    protected $prenom;

    public function __construct($nom, $prenom) {
        $this->nom = $nom;
        $this->prenom = $prenom;
    }
}

class Admin extends User {
    public function afficherAdmin() {
        return "Admin: $this->nom $this->prenom";
    }
}
```

Listing 3: Exemple d'héritage

Polymorphisme

Le polymorphisme permet à une méthode d'avoir un comportement différent selon l'objet qui l'appelle.

```php
class SuperAdmin extends Admin {
    public function afficherAdmin() {
        return "SuperAdmin: $this->nom $this->prenom";
    }
}

$admin = new Admin("Ali", "Bennani");
$superAdmin = new SuperAdmin("Fatima", "Zahra");

echo $admin->afficherAdmin(); // Admin: Ali Bennani
echo $superAdmin->afficherAdmin(); // SuperAdmin: Fatima
   Zahra
```

Listing 4: Exemple de polymorphisme

Conclusion

La POO en PHP offre de puissants outils pour modéliser des systèmes complexes en facilitant la réutilisation et la gestion du code. En combinant encapsulation, héritage et polymorphisme, les développeurs peuvent créer des applications robustes et évolutives.