



Programmation orientée objet en PHP

[Accueil](#) ▶ [Mes cours](#) ▶ [Développement logiciel](#) ▶ [POO PHP](#) ▶ [L'héritage](#) ▶ [Redéfinition de méthodes](#)

Redéfinition de méthodes

La redéfinition de méthodes permet à une sous-classe de modifier le comportement d'une méthode héritée de sa super-classe.

Dans la classe `Personne`, nous allons ajouter une méthode publique `parler()`. La classe `Formateur` héritera donc de cette méthode.

```
1 <?php
2
3 // Personne.php
4 class Personne {
5
6     protected $nom;
7     protected $prenom;
8
9     function __construct($nom, $prenom) {
10         $this->nom = $nom;
11         $this->prenom = $prenom;
12     }
13
14     public function getNom() {
15         return $this->nom;
16     }
17
18     public function getPrenom() {
19         return $this->prenom;
20     }
21
22     public function setNom($nom) {
23         $this->nom = $nom;
24     }
25
26     public function setPrenom($prenom) {
27         $this->prenom = $prenom;
28     }
29
30     public function parler() {
31         return 'je parle...';
32     }
33
34 }
```

Dans le lanceur, nous créons une instance de `Formateur` et nous appelons la méthode `parler()`.

```
1 <?php
2
3 // lanceur.php
4 include './Personne.php';
5 include './Formateur.php';
6
7 $formateur = new Formateur('Sparrow', 'Jack', 'développement logiciel');
8 echo $formateur->parler();
```

Lorsque nous appelons la méthode `parler()` avec la référence `formateur`, le comportement est bien celui défini

dans la super-classe, c'est-à-dire la classe `Personne`.

Maintenant si je souhaite donner à `Formateur` une façon de parler qui lui sera propre, je peux redéfinir la méthode `parler()` dans la classe `Formateur`.

```
1 <?php
2
3 // Formateur.php
4 class Formateur extends Personne {
5
6     private $specialite;
7
8     function __construct($nom, $prenom, $specialite) {
9         parent::__construct($nom, $prenom);
10        $this->specialite = $specialite;
11    }
12
13    public function getSpecialite() {
14        return $this->specialite;
15    }
16
17    public function setSpecialite($specialite) {
18        $this->specialite = $specialite;
19    }
20
21    public function enseigner() {
22        return 'j\'enseigne';
23    }
24
25    public function parler() {
26        return 'le PHP est un langage...';
27    }
28
29 }
```

Le lanceur n'est pas modifié. La classe `Personne` a gardé sa façon de parler, la classe `Formateur` possède une façon de parler bien à elle.

Si je souhaite ré-utiliser la méthode de la super-classe dans la méthode de la sous-classe, je peux une fois de plus utiliser le mot clé `parent` et l'opérateur `::`.

Formateur.php (modification de la méthode `parler()`)

```
1 <?php
2
3 // Formateur.php
4 class Formateur extends Personne {
5
6     private $specialite;
7
8     function __construct($nom, $prenom, $specialite) {
9         parent::__construct($nom, $prenom);
10        $this->specialite = $specialite;
11    }
12
13    public function getSpecialite() {
14        return $this->specialite;
15    }
16
17    public function setSpecialite($specialite) {
18        $this->specialite = $specialite;
19    }
20
21    public function enseigner() {
22        return 'j\'enseigne';
23    }
24 }
```

```
25     public function parler() {  
26         $msg = parent::parler();  
27         return "$msg le PHP est un langage...";  
28     }  
29  
30 }
```

Le lanceur n'est toujours pas modifié. La chaîne renvoyée par la méthode `parler()` de la classe `Personne` est assignée à la variable `msg` puis la méthode `parler()` de `Formateur` renvoie une concaténation de chaînes.

Fin

NAVIGATION



Accueil

■ Ma page

Pages du site

Mon profil

Cours actuel

POO PHP

Participants

Généralités

La programmation orientée objet : premiers pas

L'héritage

Introduction

T.P. héritage

Redéfinition de méthodes

T.P. redéfinition

Les classes abstraites

T.P. véhicule

Le mot clé final

T.P. employés

Les interfaces

Le typage

Les namespaces

Les exceptions

Les bases de données avec PDO

Les tests avec PHPUnit

Petite application version 2

Petite application version 3

[Mes cours](#)

ADMINISTRATION



Administration du cours

Réglages de mon profil

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