Introdução à POO

Introdução à Programação Orientada a Objetos (POO)

A **Programação Orientada a Objetos (POO)** é um paradigma de programação que organiza o código em torno de **objetos**. Esses objetos são instâncias de **classes**, que agrupam dados (**atributos**) e comportamentos (**métodos**) relacionados. A POO é amplamente utilizada devido à sua capacidade de organizar e reutilizar código de forma eficiente.

Benefícios da POO

A POO oferece diversas vantagens, como:

- **Modularidade**: O código pode ser dividido em partes menores e reutilizáveis.
- **Reutilização**: Classes podem ser reutilizadas em diferentes partes do programa.
- **Facilidade de manutenção**: O código fica mais organizado e fácil de modificar.
- **Extensibilidade**: Novas funcionalidades podem ser adicionadas sem alterar código existente.

Conceitos Fundamentais da POO

1. Classe

Uma **classe** é um modelo ou um molde que define as características (**atributos**) e comportamentos (**métodos**) de um objeto.

Exemplo em PHP:

```php
class Carro {
    public $marca;
    public $modelo;
```

```
    public function acelerar() {
        echo "O carro está acelerando!\n";
    }
}
```

2. Objeto

Um **objeto** é uma instância de uma classe. Ele contém atributos e pode executar métodos definidos pela classe.

Exemplo:

```
$meuCarro = new Carro();
$meuCarro->marca = "Toyota";
$meuCarro->modelo = "Corolla";
$meuCarro->acelerar();
```

Saída:

```
O carro está acelerando!
```

3. Atributos

Os **atributos** são variáveis que armazenam informações sobre o objeto. No exemplo acima, `marca` e `modelo` são atributos da classe `Carro`.

4. Métodos

Os **métodos** são funções dentro da classe que definem o comportamento dos objetos. No exemplo, o método `acelerar()` imprime uma mensagem.

5. Encapsulamento

O **encapsulamento** restringe o acesso direto a certos dados do objeto, permitindo o controle por meio de métodos específicos. Para isso, usamos modificadores de acesso:

- `public`: acessível de qualquer lugar.
- `private`: acessível apenas dentro da classe.
- `protected`: acessível dentro da classe e suas subclasses.

6. Getters e Setters

Os **getters** e **setters** são métodos usados para acessar e modificar atributos privados de uma classe.

Exemplo:

```php
class Pessoa {
    private $nome;

    public function setNome($nome) {
        $this->nome = $nome;
    }

    public function getNome() {
        return $this->nome;
    }
}

$pessoa = new Pessoa();
$pessoa->setNome("Carlos");
echo $pessoa->getNome();
```

Saída:

```
Carlos
```

7. Getters e Setters Mágicos

PHP possui métodos mágicos `__get` e `__set`, que permitem acessar e modificar atributos dinamicamente.

Exemplo:

```php
class Produto {
    private $dados = [];

    public function __set($nome, $valor) {
        $this->dados[$nome] = $valor;
    }

    public function __get($nome) {
        return $this->dados[$nome] ?? null;
    }
}

$produto = new Produto();
$produto->nome = "Celular";
echo $produto->nome;
```

Saída:

```
Celular
```

8. Construtor (`__construct`)

O **construtor** é um método especial chamado automaticamente ao instanciar um objeto.

Exemplo:

```php
class Usuario {
    public $nome;

    public function __construct($nome) {
        $this->nome = $nome;
    }
}

$usuario = new Usuario("João");
echo $usuario->nome;
```

Saída:

```
João
```

9. Herança

A **herança** permite que uma classe herde características de outra classe, evitando repetição de código.

Exemplo:

```php
class Animal {
    public function fazerSom() {
        echo "Som genérico";
    }
}

class Cachorro extends Animal {
    public function fazerSom() {
        echo "Latido";
```

```
    }
}

$dog = new Cachorro();
$dog->fazerSom();
```

Saída:

```
Latido
```

10. Polimorfismo

O **polimorfismo** permite que diferentes classes tenham métodos com o mesmo nome, mas com comportamentos distintos.

11. Namespace

Os **namespaces** ajudam a organizar o código e evitar conflitos entre classes com o mesmo nome.

Exemplo:

```
namespace Biblioteca;
class Livro {
    public function ler() {
        echo "Lendo um livro";
    }
}
```

Conclusão

A Programação Orientada a Objetos é um poderoso paradigma que ajuda a estruturar melhor os programas, facilitando a manutenção e reutilização de código. Nos próximos capítulos, exploraremos cada conceito com mais detalhes e aplicações práticas.

Este material serve como um guia inicial para quem nunca teve contato com POO, utilizando explicações claras e exemplos práticos em PHP.