INSTITUTO FEDERAL
SANTA CATARINA
Câmpus Canoinhas

Orientação a Objetos

Orientação a Objetos

O que é Orientação a Objetos?

A partir da versão 5, o PHP melhorou bastante o suporte à Programação Orientada a Objetos.

Orientação a Objetos

A OO do PHP utiliza classes e objetos. Uma definição bem chula de classe é um agrupamento de variáveis e funções, porém o conceito de OO é bem mais complexo que isso. Muitos conceitos já são explicados na disciplina de Tecnologia Orientada a Objetos, portanto eles não serão explicados novamente.

Um objeto é criado a partir de uma classe. O objeto possui todas as variáveis e funções que a sua classe. Chamaremos de atributos as variáveis de um objeto e chamaremos de métodos as funções de um objeto.

Orientação a Objetos

Vamos pensar em um exemplo bem simples: uma lâmpada. Os atributos de um objeto armazenam o seu estado atual. Quais atributos a lâmpada pode ter? Os métodos de um objeto são ações que o objeto deve realizar. Quais métodos a lâmpada pode ter?

Orientação a Objetos

Vamos ver como declarar uma classe em PHP.

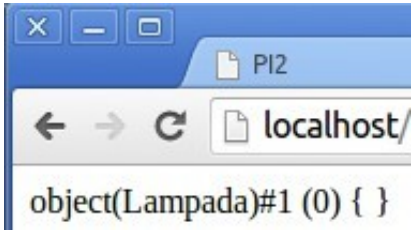
```php
class Lampada {

}
```

Podemos criar um objeto dessa classe.

```php
$objetoLampada = new Lampada();

var_dump($objetoLampada);
```



Nosso objeto não faz nada, pois ele não possui atributos e métodos.

Orientação a Objetos

Vamos criar um atributo "ligada" do tipo booleano na nossa classe Lâmpada.

```php
class Lampada {
    public $ligada = false;
}

$objetoLampada = new Lampada();
if ($objetoLampada->ligada) {
    echo 'Está ligada<br>';
} else {
    echo 'Está desligada<br>';
}
var_dump($objetoLampada);
```

Para acessar atributos ou métodos de um objeto, deve ser utilizada flecha >.

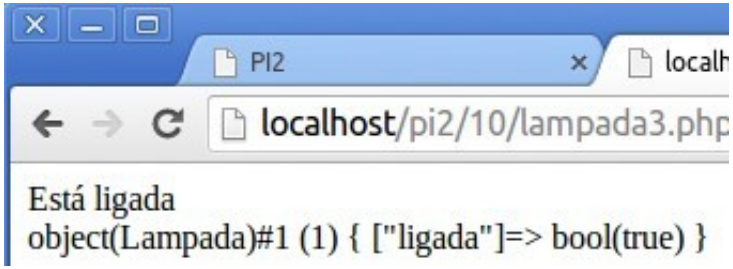
Orientação a Objetos

Vamos criar um método para ligar a lâmpada.

```php
class Lampada {
    public $ligada = false;

    function ligar() {
        $this->ligada = true;
    }
}

$objetoLampada = new Lampada();
$objetoLampada->ligar();
if ($objetoLampada->ligada) {
    echo 'Está ligada<br>';
} else {
    echo 'Está desligada<br>';
}
var_dump($objetoLampada);
```



Para acessar um atributo ou método do objeto de dentro do objeto, **deve** ser utilizado $this >.

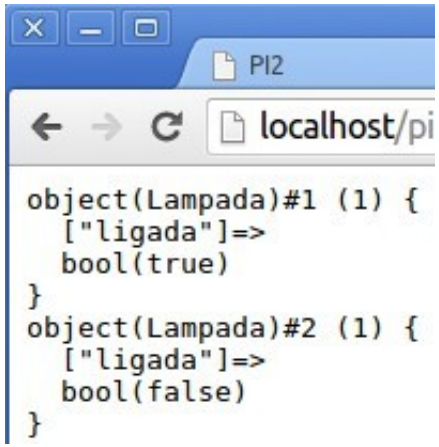
Orientação a Objetos

Podemos criar quantos objetos quisermos da mesma classe. Cada objeto é independente do outro.

```php
class Lampada {
    public $ligada = false;

    function ligar() {
        $this->ligada = true;
    }
}

$lampada1 = new Lampada();
$lampada1->ligar();
$lampada2 = new Lampada();
echo '<pre>';
var_dump($lampada1);
var_dump($lampada2);
```



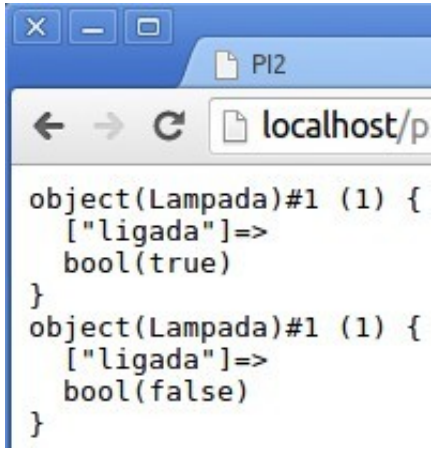
Orientação a Objetos

Não podemos somente ligar uma lâmpada. Também precisamos desligar ela.

```php
class Lampada {
    public $ligada = false;

    function ligar() {
        $this->ligada = true;
    }
    function desligar() {
        $this->ligada = false;
    }
}

$lampada = new Lampada();
$lampada->ligar();
var_dump($lampada);
$lampada->desligar();
var_dump($lampada);
```

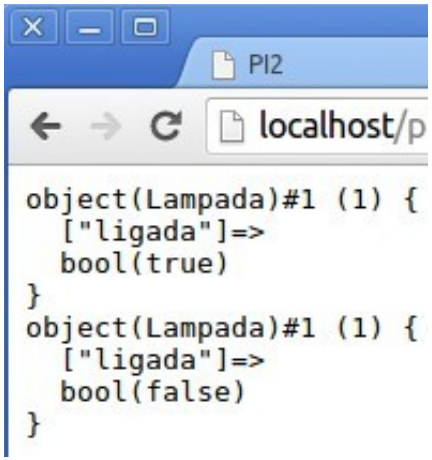


Orientação a Objetos

Já que o atributo da classe possui o modificador "public", o atributo pode ser acessado "fora" da classe.

```php
class Lampada {
    public $ligada = false;
}

$lampada = new Lampada();
$lampada->ligada = true;
var_dump($lampada);
$lampada->ligada = false;
var_dump($lampada);
```



Não é interessante deixar os atributos públicos, pois quebra o encapsulamento.

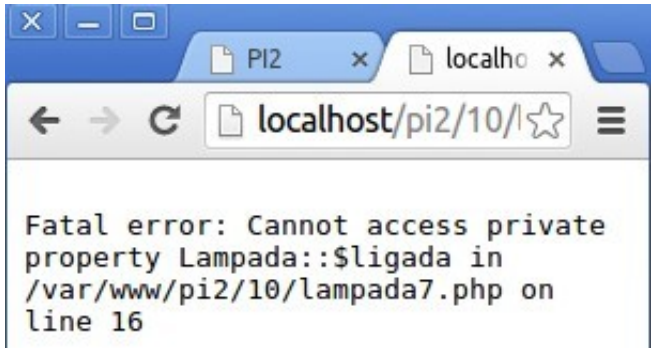
Orientação a Objetos

Modificamos o atributo para "private". Se tentarmos acessar o atributo de "fora" da classe, ocorrerá um erro.

```php
class Lampada {
    private $ligada = false;

    function ligar() {
        $this->ligada = true;
    }
    function desligar() {
        $this->ligada = false;
    }
}

$lampada = new Lampada();
$lampada->ligada = true;
```



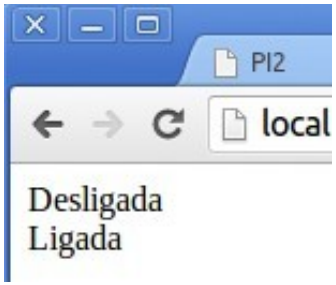
Orientação a Objetos

Os atributos por padrão devem ser privados. Para ler ou escrever um atributo, devemos utilizar métodos!

```php
class Lampada {
    private $ligada = false;

    function ligar() {
        $this->ligada = true;
    }
    function desligar() {
        $this->ligada = false;
    }
    function estaLigada() {
        return $this->ligada;
    }
}

$lampada = new Lampada();
echo $lampada->estaLigada() ? 'Ligada<br>' : 'Desligada<br>';
$lampada->ligar();
echo $lampada->estaLigada() ? 'Ligada<br>' : 'Desligada<br>';
```

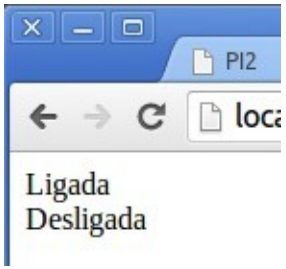


Orientação a Objetos

Estamos imprimindo toda hora o estado da lâmpada. Para facilitar o nosso trabalho, podemos criar um método que faz isso.

```php
class Lampada {
    private $ligada = false;

    function ligar() {
        $this->ligada = true;
    }
    function desligar() {
        $this->ligada = false;
    }
    function estaLigada() {
        return $this->ligada;
    }
    function imprimirEstado() {
        echo $this->ligada ? 'Ligada<br>':'Desligada<br>';
    }
}
$lampada = new Lampada();
$lampada->ligar();
$lampada->imprimirEstado();
$lampada->desligar();
$lampada->imprimirEstado();
```

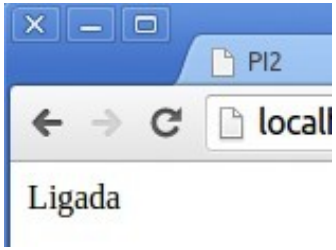


Orientação a Objetos

A sempre é criada desligada. Mas tem como criar ela ligada? Devemos utilizar um construtor para isso.

```php
class Lampada {
    private $ligada = false;

    function __construct($ligada) {
        $this->ligada = $ligada;
    }
    function estaLigada() {
        return $this->ligada;
    }
    function imprimirEstado() {
        echo $this->ligada ? 'Ligada<br>':'Desligada<br>';
    }
}
$lampada = new Lampada(true);
$lampada->imprimirEstado();
```



O método especial **construct** é chamado quando um objeto é criado.

Orientação a Objetos

Este tipo de construtor existe a partir da versão 5 do PHP. Nas versões anteriores, o construtor era um método que possuía o mesmo nome que a classe.

A partir da versão 5, vocês devem utilizar o método **__construct** para definir um construtor.

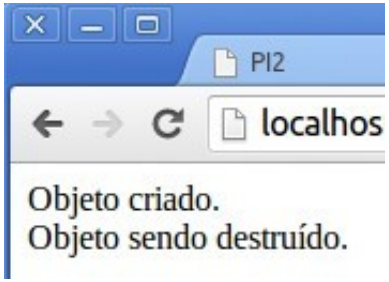
Análogamente ao construtor, a classe também pode ter um destrutor. Ele é chamado quando o objeto é destruído. Normalmente o destrutor é utilizado em casos que precisem ser liberados recursos travados pelo objeto.

Orientação a Objetos

O método destrutor é o **destruct**.

```php
class Lampada {
    private $ligada = false;

    function __construct() {
        echo 'Objeto criado.<br>';
    }
    function __destruct() {
        echo 'Objeto sendo destruído.<br>';
    }
}
$lampada = new Lampada();
unset($lampada);
```



Normalmente, o método construtor é muito utilizado e o destrutor é pouco utilizado.

Orientação a Objetos

Existem várias convensões que devem ser seguidas com relação aos identificadores.

O **nome de classe** deve sempre ter a primeira letra maiúscula. Se o nome tiver várias palavras, então a primeira letra de cada palavra deve ser maiúscula. As outras letras são minúsculas.

Nome de atributo segue a convensão de nome de variável. É a mesma coisa que o nome de classe, porém a primeira letra é minúscula.

Nome de método segue a mesma convensão de nome de função, que é a mesma para nomes de variáveis.

Orientação a Objetos

Também existe uma convensão para métodos acessores (métodos para ler e escrever em um atributo).

Se o método escreve em um atributo, ele deve ser chamado **setNomeDoAtributo**. Este método possui um parâmetro e deve atribuir o valor do parâmetro no atributo.

Se o método lê um atributo, ele deve ser chamado **getNomeDoAtributo**. Este método não possui parâmetros e deve retornar o valor do atributo. Além do "get...", caso o atributo seja do tipo booleano, o método pode ser chamado de **isNomeDoAtributo**.

Orientação a Objetos

Exemplo de métodos acessores.

```php
class Lampada {
    private $ligada = false;
    private $material = 'tungsténio';

    function setMaterial($material) {
        $this->material = $material;
    }
    function getMaterial() {
        return $this->material;
    }
    function setLigada($ligada) {
        $this->ligada = $ligada;
    }
    function isLigada() {
        return $this->ligada;
    }
}
```

Exercício

Crie uma classe "Pessoa". Ela deve ter os atributos: nome (string), idade (inteiro) e sexo_masculino (booleano). Ela deve ter 2 métodos. O primeiro é o método construtor. Ele deve receber um valor para cada atributo como parâmetro. O segundo método deve imprimir os dados da pessoa. Crie pelo menos 2 objetos da classe Pessoa e teste o método imprimir.

Referências

NIEDERAUER, Juliano. **Desenvolvendo websites com PHP**. 2. ed. São Paulo: Novatec, 2011. 301 p. ISBN 9788575222348.