

Orientação a Objetos com PHP

Wendell Bento Geraldles

Instituto Federal de Goiás campus Luziânia

24 de junho de 2019

Sumário

- 1 Encapsulamento
- 2 Private
- 3 Protected
- 4 Public
- 5 Referências bibliográficas

Orientação a Objetos com PHP

Encapsulamento

O encapsulamento é um mecanismo que provê proteção de acesso aos membros internos de um objeto. Existem certas propriedades de uma classe que devem ser tratadas exclusivamente por métodos dela mesma, que são projetadas para manipular essas propriedades de forma correta. (DALL'OGGIO, 2007)

Figura: Encapsulamento

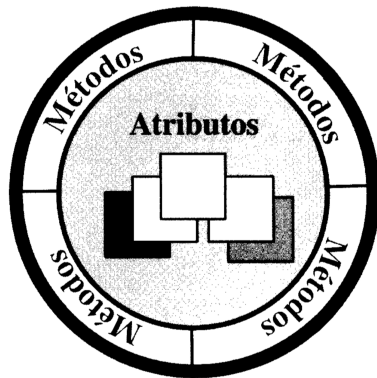


Figura 2.6 – Encapsulamento.

Para atingir o encapsulamento, uma das formas é definindo a visibilidade das propriedades de um objeto. A visibilidade define a forma como essas propriedades devem ser acessadas. Existem três formas de acesso: (DALL'OGGIO, 2007)

Visibilidade	Descrição
private	Membros declarados como private somente podem ser acessados dentro da própria classe em que foram declarados. Não poderão ser acessados a partir de classes descendentes nem a partir do programa que faz uso dessa classe.
protected	Membros declarados como protected somente podem ser acessados dentro da própria classe em que foram declarados e a partir de classes descendentes, mas não a partir do programa que faz uso dessa classe.
public	Membros declarados como public poderão ser acessados livremente a partir da própria classe em que foram declarados, a partir de classes descendentes e a partir do programa que faz uso dessa classe.

Orientação a Objetos com PHP

Private

Para demonstrar a visibilidade private, criaremos a classe Funcionario e marcaremos a maioria das propriedades como private. A única propriedade que deixaremos para livre acesso será o nome. (DALL'OGGIO, 2007)

Código-Fonte 1: Funcionario.class.php

```
<?php
class Funcionario {
    private $Codigo;
    public $Nome;
    private $Nascimento;
    private $Salario ;
}
?>
```

Orientação a Objetos com PHP

Private

Código-Fonte 2: private.php

```
<?php  
include_once 'classes/Funcionario.class.php'; //insere a classe  
$pedro = new Funcionario; //cria um objeto  
$pedro->Salario = 'Oitocentos e setenta e seis';  
?>
```

Orientação a Objetos com PHP

Private

Veja que o PHP resulta em um erro de acesso à propriedade, como esperado, mas se não podemos alterar o valor da propriedade Salario dessa forma, como faremos? Criaremos métodos pertencentes à classe Funcionário para manipular essa propriedade. (DALL'OGGIO, 2007)

Código-Fonte 3: Funcionario.class.php

```
<?php
class Funcionario {
    private $Codigo;
    public $Nome;
    private $Nascimento;
    private $Salario ;
}
?>
```

Orientação a Objetos com PHP

Private

Código-Fonte 4: Funcionario.class.php

```
<?php
class Funcionario {
    // atribui o parametro $Salario a propriedade $Salario
    function SetSalario ( $Salario ) {
        if ( is_numeric( $Salario ) and ( $Salario > 0 ) ) {
            $this->Salario = $Salario;
        }
    }
}
function GetSalario () {
    return $this->Salario; // retorna o valor da propriedade $Salario
}
?>
```


Orientação a Objetos com PHP

Private

Código-Fonte 5: private.php

```
<?php
include_once 'classes/Funcionario.class.php';
// instancia um novo funcionario
$pedro = new Funcionario;
// atribui novo Salario
$pedro->SetSalario(876);
//obtem o Salario
echo ' Salario : (R$) ' . $pedro->GetSalario();
?>
```

Orientação a Objetos com PHP

Protected

Para demonstrar a visibilidade **protected**, vamos especializar a classe Funcionario, criando a classe Estagiario. A única característica exclusiva de estagiário é que o seu salário é acrescido de 12% de bônus. Para tanto, sobrescreveremos o método **GetSalario()** para que este retorne o salário já reajustado. (DALL'OGGIO, 2007)

Código-Fonte 6: Estagiario.class.php

```
<?php
class Estagiario extends Funcionario {
    function GetSalario() {
        return $this->Salario * 1.12;
    }
}
```

Orientação a Objetos com PHP

Protected

Código-Fonte 7: protected.php

```
<?php
include_once 'classes/Funcionario.class.php';
include_once 'classes/Estagiario.class.php';
$pedrinho = new Estagiario;
$pedrinho->SetSalario(248);
echo 'O Salario de Pedrinho e R$: ' . $pedrinho->GetSalario() . "\n";
?>
```

Orientação a Objetos com PHP

Protected

O programa não irá retornar o salário esperado. Isso ocorre por que a propriedade Salario é uma propriedade private, o qe significa que ela somente pode ser acessada de dentro da classe em que ela foi declarada (Funcionario). Para permitir o acesso também nas classes-filha, alteramos a classe Funcionario e marcaremos a propriedade Salario como protected. (DALL'OGGIO, 2007)

Código-Fonte 8: Funcionario.class.php

```
<?php
class Funcionario {
    private $Codigo;
    public $Nome;
    private $Nascimento;
    protected $Salario ;

}
```

```
?>
```

Orientação a Objetos com PHP

Protected

Código-Fonte 9: protected.php

```
<?php
include_once 'classes/Funcionario.class.php';
include_once 'classes/Estagiario.class.php';
$pedrinho = new Estagiario;
$pedrinho->SetSalario(248);
echo 'O Salario de Pedrinho e R$: ' . $pedrinho->GetSalario() . "\n";
?>
```

Orientação a Objetos com PHP

Public


Demonstrar a visibilidade public é uma tarefa simples, pois o comportamento padrão do PHP é tratar uma propriedade como public. No exemplo a seguir, estamos criando dois objetos e alterando suas propriedades à vontade, as quais poderiam ser alteradas por métodos internos e por classes descendentes também. (DALL'OGGIO, 2007)

Orientação a Objetos com PHP

Public

Código-Fonte 10: public.php

```
<?php
include_once 'classes/Funcionario.class.php';
include_once 'classes/Estagiario.class.php';
$pedrinho = new Funcionario;//cria um objeto Funcionario
$pedrinho->nome = 'Pedrinho';
$mariana = new Estagiario;//cria um objeto Estagiario
$mariana->nome = 'Mariana';
echo $pedrinho->nome;//imprime a propriedade nome
echo $mariana->nome;//imprime a propriedade nome
?>
```

 DALL'OGGIO, P. **PHP: programando com orientação a objetos**. 1. ed. São Paulo: Novatec Editora, 2007. 580 p.