

PÓS-GRADUAÇÃO

Programação PHP



PHP e Orientação a Objetos

Bloco 1


Thiago Salhab Alves





► PHP e orientação a objetos

Objetivos:

- Aprender os conceitos de orientação a objetos.
 - Aprender a utilizar a orientação a objetos no PHP.
 - Aprender sobre controle de exceções no PHP.
- 



► Orientação a objetos

- Segundo Saraiva e Barreto (2018), a orientação a objetos (OO) é o desenvolvimento de uma estratégia em que os sistemas devem ser construídos baseados em uma coleção de componentes reutilizáveis, conhecidos como objetos, tomando por definição que objetos possuem basicamente duas características:
- Objetos possuem dados.
- Objetos manipulam dados e fazem coisas.



► Orientação a objetos

Objeto

É a entidade que se deseja generalizar (torná-la uma classe) e pode ser uma pessoa, um lugar, um evento, um conceito, um relatório, uma tela, ou seja, qualquer coisa real.



► Orientação a objetos

Classe

- Uma classe é a abstração de um objeto, ou seja, é um modelo a partir do qual os objetos podem ser criados.
- Enquanto um objeto representa uma coisa real e somente uma coisa em um determinado instante, uma classe é a generalização deste, mostrando quais devem ser suas características (dados) e quais as suas capacidades (funcionalidades).

► Orientação a objetos

Atributo

- É cada uma das peças de dados de uma classe, ou seja, a coleção de atributos representa o que uma classe sabe (os dados que a classe possui).
- Em termos de programação, os atributos são as variáveis que definimos para a classe, e que são utilizadas apenas na classe.



► Orientação a objetos

Método

- Os métodos definem o que as classes sabem fazer, ou seja, os métodos podem alterar os atributos e realizar funções inerentes à classe (por exemplo, incluir um novo funcionário, alterar seu salário, etc.).
- Em termos de programação pode-se pensar em métodos como funções na classe, as quais podem retornar dados (o salário do funcionário) ou não, conforme suas características funcionais.



► Orientação a objetos

Herança

- É a capacidade de uma classe herdar os atributos e métodos de uma outra classe. Chamamos de superclasse a classe mãe, aquela que fornece os atributos e métodos à subclasse que os herda.

► Orientação a objetos no PHP

Figura 1 – Exemplo de classe no PHP

```
<?php
class Aluno {
    public $ra;
    public $nome;
    public $curso;

    function getCurso(){
        return $this->curso;
    }
    function setCurso($cur){
        $this->curso = $cur;
    }
}
?>
```

Fonte: elaborada pelo autor.

► Orientação a objetos no PHP

Figura 2 – Exemplo de instanciação de classe no PHP

```
<?php
    $_c = new carro();
    $_c ->
setMarca("Toyota"):
    $_c ->
setModelo("Corolla");
    $_c -> setcor("Preto");
    $_c -> setAno (2019);
    echo $_c -> getMarca();
    echo "<br/>";
    echo $_c -> getCarro();
?
```

Fonte: elaborada pelo autor.

Orientação a Objetos no PHP

Bloco 2

Thiago Salhab Alves





► Orientação a objetos no PHP

- Segundo Soares (2013), a partir da versão 5 do PHP, os recursos de orientação a objetos foram incorporados.
- Estão disponíveis recursos existentes em outras linguagens fortemente orientadas a objetos como o Java.
- Recursos como gerenciamento de exceções (*try, catch*), herança, definição de atributos e métodos públicos e privados, construtores, destruidores e reflexão de classes, permitindo uma melhor estruturação dos sistemas em PHP.

► Orientação a objetos no PHP

Figura 3 – Exemplo de atributo público no PHP

```
<?php
class aluno {
    public $ra;
    public function setRa($r) {
        $this->ra = $r;
    }
}
$a = new aluno();
$a->setRA("12345");
$a->ra = "67890";
?>
```

Fonte: elaborada pelo autor.

► Orientação a objetos no PHP

Figura 4 – Exemplo de atributo protegido no PHP

```
<?php
class aluno {
    protected $ra;
    public function setRa($r) {
        $this->ra = $r;
        echo "Ra: $this->ra";
    }
}

class graduacao extends aluno{
    protected $tcc;

    public function setTCC($t){
        $this->tcc = $t;
        echo "TCC: $this->tcc";
    }
}
```

```
$a = new aluno();
$a->setRA("12345");
$a = new graduacao();
$a->setRa("67890");
$a->setTCC("Programação Web");
?>
```

Fonte: elaborada
pelo autor.

► Orientação a objetos no PHP

Figura 5 – Exemplo de atributo privado no PHP (continua)

```
<?php
class pessoas {
    private $_tipo;
    protected $_nome;
    protected $_endereço;
    protected $_telefone;

    protected function setTipo($_t) {
        $this->_tipo = $_t;
    }
}
```

Fonte: elaborada pelo autor.

► Orientação a objetos no PHP

Figura 6 – Exemplo de atributo privado no PHP (continuação)

```
class estudante extends pessoas {  
    protected $_curso;  
    function __CONSTRUCTOR(){  
        parent::setTipo("E");  
    }  
}  
  
$e = new estudante();  
$e->_tipo = "A"; // Erro  
?>
```

Fonte: elaborada pelo autor.

► Orientação a objetos no PHP

Construtor e destruidor de classes

- O construtor, segundo Soares (2013), referenciado no PHP como `__construct()`, é uma função definida na classe e que é executada sempre que o objeto é criado (isto é, sempre que a classe é instanciada).
- O destruidor da classe, que é definido por meio da função `__destruct()`, é executado sempre que o objeto for destruído.

PÓS-GRADUAÇÃO

Teoria em prática

Bloco 3

Thiago Salhab Alves





► Teoria em prática

Você é um programador PHP e está escrevendo seu novo código para um projeto, e se deparou com a necessidade de reorganizar seu código no paradigma orientado a objetos. Você já havia escrito seu código para cadastro de clientes de forma estruturada, e agora gostaria de reescrever com o paradigma orientado a objetos. Porém, ficou na dúvida de como realizar essa reescrita e quais elementos utilizar. Como podemos fazer a reescrita de uma classe, com atributos e métodos, para cadastro de clientes?

► Teoria em prática

Figura 7 – Reescrita de uma classe, com atributos e métodos, para cadastro de clientes (continua)

```
<?php
class cliente {
    private $_código, $_nome, $RG,
    $CPF, $_Tel;
    public function
    setCodigo($_cod){
        $this->_código = $_cod;
    }
    public function
    setNome($_nome){
        $this->_nome = $_nome;
    }
    public function setRG($_rg){
        $this->_RG = $_rg;
    }
}
```

Fonte: elaborada pelo autor.

► Teoria em prática

Figura 8 – Reescrita de uma classe, com atributos e métodos, para cadastro de clientes (continuação)

```
public function setCPF($_cpf){
    $this->_CPF = $_cpf;
}

public function setTel($_tel){
    $this->_Tel = $_tel;
}

public function getDados () {
    return "Código: " . $this->_código .
    "<br/>". "Nome: " . $this->_nome . "<br/>".
    "RG: " . $this->_RG .
    "<br/>". "CPF: " . $this->_CPF .
    "<br/>". "Tel: " . $this->_Tel;
}
}
?>
```

Fonte: elaborada pelo autor.

PÓS-GRADUAÇÃO

Dica do professor

Bloco 4

Thiago Salhab Alves



► Dica do professor

Leitura de capítulo de livro:

- Leitura de artigo da base EBSCO (BV). PAVLAK, Mike. Manage PHP Erros Effectively. **iPro Developer**, [s.d.], v. 2, 6. ed., p, 29-33, jun./2013.
- Leitura do capítulo 13 (*PHP e OO*). SOARES, Walace. **PHP 5: Conceitos, Programação e Integração com Banco de Dados**. 7. ed. São Paulo: Érica, 2013.



► Referências

SARAIVA, Maurício de O.; BARRETO, Jeanine dos S. **Desenvolvimento de Sistemas como PHP**. Porto Alegre: SAGAH, 2018.

SOARES, Walace. **PHP 5: Conceitos, Programação e Integração com Banco de Dados**. 7. ed. São Paulo: Érica, 2013.

MILETTO, Evandro M.; BERTAGNOLLI, Silvia de C. **Desenvolvimento de Software II: Introdução ao Desenvolvimento Web como HTML, CSS, JavaScript e PHP**. Porto Alegre: Bookman, 2014.

