PHP Orientado a Objetos

Universidade Estadual de Santa Cruz - UESC
Disciplina: Introdução a programação WEB
Autor: Patrick Silva Ferraz
Orientador: Prof. Dany D. Sanchez

Autor: Patrick Silva Ferraz

Legenda

- Vermelho: Destaque ao título do tópico;
- Azul: Início e fim de código php e nome de classes, variáveis, métodos;
- Verde: Palavras reservadas da linguagem;
- Magenta: Campos de escolha do programador;
- entre aspas “duplas”/’simples’: Strings e/ou citações;
- Cinza: observações, iniciado com // e *Itálico* representa c*omentários;*

Autor: Patrick Silva Ferraz

Roteiro

- **Pilares da POO**
  - Abstração
    - *Classes; Propriedades; Métodos.*
  - Encapsulamento
    - *Visibilidade; Constantes; Estáticas; Operador (::); Final.*
  - Herança
    - *Extends; Abstract; Interface; Traits.*
  - Polimorfismo
    - *Instanceof; Sobreposição (Override).*
- **Informações úteis**
  - Sobrecarga (overload)
  - Namespaces
  - Classes anônimas
  - Funções para Classes/Objetos
  - Comparando Objetos
  - Constantes Mágicas

Autor: Patrick Silva Ferraz

Pilares da POO

- Abstração
- Encapsulamento
- Herança
- Polimorfismo

Autor: Patrick Silva Ferraz

Pilares da POO

- **Abstração**
- Encapsulamento
- Herança
- Polimorfismo

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração

- **Abstração (representação de um objeto real)**
  - Identidade: classes
    - Ex: cachorro, gato...
  - Características: propriedades (variáveis)
    - Ex: tamanho, raça, idade...
  - Ações: métodos (funções)
    - Ex: latir, correr...

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração

- Abstração (representação de um objeto real)
  - **Classes**
  - Propriedades
  - Métodos

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração - Classe

- Estruturando Classes (class)

```php
<?php
    class ExemploClasse
    {
        //propriedades
        //métodos
    }
?>
```

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração - Classe

- Instanciando a classe

```
<?php
    $obj = new ExemploClasse;
?>
```

- Visualizando conteúdo da classe

```
<?php
    var_dump($obj);
?>
```

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração

- Abstração (representação de um objeto real)
  - Classes
  - **Propriedades**
  - Métodos

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração - Propriedades

- Estruturando propriedades

```php
<?php
    class ExemploPropriedade
    {
        //declaração de propriedade
        public $prop = “sou uma propriedade”;
    }
?>
```

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração - Propriedades

- Instanciando a classe

```
<?php
    $obj = new ExemploPropriedade;
?>
```

- Acessando a propriedade

```
<?php
    $obj->prop;
?>
```

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração

- Abstração (representação de um objeto real)
  - Classes
  - Propriedades
  - **Métodos**

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração - Métodos

- Estruturando métodos (function)

```php
<?php
    class ExemploMetodo
    {
        //declaração de método
        public function primeiroMetodo(){
            echo “primeiro metodo”;
        }
    }
?>
```

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração - Métodos

- Instanciando a classe

```php
<?php
    $obj = new ExemploMetodo;
?>
```

- Acessando métodos

```php
<?php
    $obj->primeiroMetodo();
?>
```

Cod.: /srv/www/htdocs/dev/docsPHPOO/01_exemplo_abstracao.php

Exec.: http://localhost/dev/docsPHPOO/01_exemplo_abstracao.php

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração - Métodos

- **Pseudo-variável $this**
  - Disponível quando um método é chamado a partir de um contexto de objeto (referência ao objeto).

Cod.: /srv/www/htdocs/dev/docsPHPOO/03_exemplo_metodo_pseudo_variavel_this.php

Exec.: http://localhost/dev/docsPHPOO/03_exemplo_metodo_pseudo_variavel_this.php

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração - Métodos

- **Métodos mágicos**
  - Métodos especiais chamados quando certas ações comuns ocorrem com objetos

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração - Métodos

- Métodos mágicos mais comuns
  - __construct()
  - __destruct()
  - __toString()

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração - Métodos

- Método mágico __construct()
  - Método chamado a cada objeto instanciado.
  - Até a versão PHP 4 construtores possuíam o mesmo nome da classe ao qual pertencia (obsoleto a partir do PHP7).

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração - Métodos

- Método mágico __destruct()
  - Chamado quando todas referências a um objeto particular forem removidas ou quando o objeto for explicitamente destruído.

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração - Métodos

- Método mágico __toString()
  - Evita erros caso o script tente mostrar a classe como uma string.

Cod.: /srv/www/htdocs/dev/docsPHPOO/03_exemplo_metodo_magico_construct_destruct_toString.php

Exec.: http://localhost/dev/docsPHPOO/03_exemplo_metodo_magico_construct_destruct_toString.php

Autor: Patrick Silva Ferraz

Pilares da POO: Abstração - Métodos

- Outros métodos mágicos
  - __call()
  - __callStatic()
  - __get()
  - __set()
  - __isset()
  - __unset()
  - __sleep()
  - __wakeup()
  - __invoke()
  - __set_state()
  - __clone()
  - __debugInfo()

Infor.: https://secure.php.net/manual/pt_BR/language.oop5.magic.php#object.tostring

Autor: Patrick Silva Ferraz

Paradigma POO – 4 pilares

- Abstração
- **Encapsulamento**
- Herança
- Polimorfismo

Autor: Patrick Silva Ferraz

Pilares da POO: Encapsulamento

- **Encapsulamento**
  - Elementos que adicionam segurança à aplicação pelo fato de esconder as propriedades.

Autor: Patrick Silva Ferraz

Pilares da POO: Encapsulamento - visibilidade

- **Visibilidade**
  - public
    - Podem ser acessados de qualquer local
  - protected
    - Somente acessados em suas classes declarantes e suas classes herdeiras
  - private
    - Somente acessados em suas classes declarantes.

Cod.: /srv/www/htdocs/dev/docsPHPOO/04_exemplo_encapsulamento_visibilidade.php

Exec.: http://localhost/dev/docsPHPOO/04_exemplo_encapsulamento_visibilidade.php

Autor: Patrick Silva Ferraz

Pilares da POO: Encapsulamento

- Constantes
- Estáticas
- Operador (::)
- Final

Autor: Patrick Silva Ferraz

Pilares da POO: Encapsulamento

- **Constantes**
  - Pode ser definido em cada classe permanecendo imutável
  - Não utiliza o símbolo $ para declará-las ou usá-las
  - São alocadas por classe, e não em cada instância de classe

Autor: Patrick Silva Ferraz

Pilares da POO: Encapsulamento

- **Estáticas**
  - Propriedades e métodos estáticos podem ser acessados sem a necessidade de instanciar a classe.
  - Propriedades estáticas não podem ser acessadas através do operador ->.

Autor: Patrick Silva Ferraz

Pilares da POO: Encapsulamento

- **Operador de resolução de escopo (::)**
  - Chamado de Paamayim Nekudotayim
  - Permite acesso a métodos ou propriedades estáticas, constantes, e sobrecarregadas de uma classe.
  - Palavras-chaves:
    - self, parent, static – Utilizadas para acessar propriedades e métodos dentro de uma definição de classe.
    - nome_class ou $variável com nome_class – Utilizadas para acesso externo de uma definição de classe.

Cod.: /srv/www/htdocs/dev/docsPHPOO/05_exemplo_constante_estatica.php

Exec.: http://localhost/dev/docsPHPOO/05_exemplo_constante_estatica.php

Autor: Patrick Silva Ferraz

Pilares da POO: Encapsulamento

- **Final**
  - Impede que classes filhas sobrescrevam um método que esteja prefixado sua definição com final.
  - Se a classe estiver definida como final, ela não pode ser estendida.
  - Propriedades não podem ser declaradas como finais.

Cod.: /srv/www/htdocs/dev/docsPHPOO/06_exemplo_final.php

Exec.: http://localhost/dev/docsPHPOO/06_exemplo_final.php

Autor: Patrick Silva Ferraz

Paradigma POO – 4 pilares

- Abstração
- Encapsulamento
- **Herança**
- Polimorfismo

Autor: Patrick Silva Ferraz

Pilares da POO: Herança

- **Herança**
  - Provê reuso de código.
  - A subclasse herda todos métodos públicos e protegidos da classe pai.
  - Utilização da palavra reservada extends
    - Ex: class NovaClasse extends ClassePai{}

Autor: Patrick Silva Ferraz

Pilares da POO: Herança - Abstração

- **Abstração de classes**
  - Classes abstratas não podem ser instanciadas.
  - Classes com ao menos um método abstrato também devem ser abstratas.

Autor: Patrick Silva Ferraz

Pilares da POO: Herança - Abstração

- **Abstração de classes**
  - Ao herdar uma classe abstrata, todos os métodos abstratos da classe pai devem ser implementados nas classes filhas com a mesma visibilidade (ou menos restrita).

Autor: Patrick Silva Ferraz

Pilares da POO: Herança - Abstração

- **Abstração de classes**
  - A assinatura dos métodos abstratos devem coincidir, inclusive o número de argumentos (é possível definir argumentos opcionais).
  - Utilização da palavra reservada abstract
    - Classe:

      ```
      abstract class NovaClasse{}
      ```

    - Método:

      ```
      abstract public/protected function metodo();
      ```

Autor: Patrick Silva Ferraz

Pilares da POO: Herança - Interface

- **Interfaces de objetos**
  - Especificam quais métodos uma classe deve implementar.
  - Nenhum método possui conteúdo definido.
  - Todos métodos da interface devem ser públicos.

Autor: Patrick Silva Ferraz

Pilares da POO: Herança - Interface

- **Interfaces de objetos**
  - Classes podem implementar mais de uma interface, separando cada interface com uma vírgula (métodos duplicados devem possuir mesma assinatura).
  - Interfaces podem ser estendidas usando a palavra extends.

Autor: Patrick Silva Ferraz

Pilares da POO: Herança - Interface

- **Interfaces de objetos**
  - É possível ter constantes em interfaces, com exceção de não poderem ser sobrescritas.

Autor: Patrick Silva Ferraz

Pilares da POO: Herança - Interface

- **Interfaces de objetos**
  - Utilização
    - Estrutura (interface):

      ```
      interface ClasseInterface{}
      ```

    - Implementar um interface (implements):

      ```
      class UmaClasse implements ClasseInterface{}
      ```

Cod.: /srv/www/htdocs/dev/docsPHPOO/07_exemplo_heranca.php

Exec.: http://localhost/dev/docsPHPOO/07_exemplo_heranca.php

Autor: Patrick Silva Ferraz

Pilares da POO: Herança - Traits

- **Traits**
  - Mecanismo para reutilização de código em linguagens de herança única.
  - Permite utilização livremente em várias classes independente da herança.
  - Não é possível instanciar.
  - Possui o mesmo comportamento de uma classe;

Cod.: /srv/www/htdocs/dev/docsPHPOO/08_exemplo_heranca_traits.php

Exec.: http://localhost/dev/docsPHPOO/08_exemplo_heranca_traits.php

Autor: Patrick Silva Ferraz

Paradigma POO – 4 pilares

- Abstração
- Encapsulamento
- Herança
- **Polimorfismo**

Autor: Patrick Silva Ferraz

Pilares da POO: Polimorfismo

- **Polimorfismo**
  - Alteração do funcionamento interno de um método herdado de um objeto pai.

Autor: Patrick Silva Ferraz

Pilares da POO: Polimorfismo

- **Operador instanceof**
  - Usado para determinar se um objeto é instancia de uma certa classe.
  - Utilização:

```
function funcao($a instanceof NomeClasse){}
function funcao(NomeClasse $a){}
```

Autor: Patrick Silva Ferraz

Pilares da POO: Polimorfismo - override

- **Sobreposição (override)**
  - Basta redeclarar o método ou a propriedade.
  - É possível preservar a funcionalidade originais de um método sobreposto com a palavra reservada parent juntamente ao operador de resolução de escopo (::).

Cod.: /srv/www/htdocs/dev/docsPHPOO/09_exemplo_sobreposicao.php

Exec.: http://localhost/dev/docsPHPOO/09_exemplo_sobreposicao.php

- **Informações adicionais**

Autor: Patrick Silva Ferraz

Informaçoes adicionais: Sobrecarga

- **Sobrecarga**
  - Diferente da maioria das linguagens OO.
  - Provê recursos para “criar” dinamicamente propriedades e métodos.

Autor: Patrick Silva Ferraz

Informações adicionais: Sobrecarga

- **Sobrecarga**
  - Invocados ao interagir com propriedades ou métodos que não foram declarados ou não são visíveis no escopo corrente.
  - Todos métodos de sobrecarga devem ser definidos como públicos.

Autor: Patrick Silva Ferraz

Informações adicionais: Sobrecarga

- **Sobrecarga de propriedades**
  - __set()
  - __get()
  - __isset()
  - __unset()
- **Sobrecarga de métodos**
  - __call()
  - __callStatic()

Infor.: https://secure.php.net/manual/pt_BR/language.oop5.overloading.php

Autor: Patrick Silva Ferraz

Informações adicionais: Classes anônimas

- **Classes anônimas**
  - Úteis quando objetos descartáveis precisarem ser criados
  - Utilização:

```php
<?php
    function funcao(){
        return new class{//métodos e propriedades}
    }
?>
```

Autor: Patrick Silva Ferraz

Informações adicionais: Funções para Classes/Objetos

- **Funções para Classes/Objetos**
  - class_exists - Checa se uma classe foi definida. O mesmo serve para:
    - method_exists e trait_exists
  - get_declared_classes - Retorna um array com todas classes definidas. O mesmo serve para:
    - get_declared_interfaces e get_declared_traits

Infor.: https://secure.php.net/manual/pt_BR/ref.classobj.php

Autor: Patrick Silva Ferraz

Informações adicionais: Função autoload

- **Função** spl_autoload_register() **(**__autoload**)**
  - Permite que classes e interfaces sejam automaticamente carregadas se elas ainda não foram definidas

Autor: Patrick Silva Ferraz

Informações adicionais: Função autoload

- **Função** spl_autoload_register() **(**__autoload**)**
  - Utilização:

```
<?php
    spl_autoload_register(function($class_name){
        include $class_name . ".php";
    });

    $obj = new MyClass();
?>
```

Autor: Patrick Silva Ferraz

Informações adicionais: Namespaces

- **namespaces**
  - Evita “colisões de nomes” (quando classes, funções, variáveis ou constantes com o mesmo nome);
  - namespaces nomeProjeto; *//código*
  - namespaces nomeProjeto {*//código*}

Cod.: /srv/www/htdocs/dev/docsPHPOO/10_exemplo_namespace.php

Cod.exec: /srv/www/htdocs/dev/docsPHPOO/10_exemplo_namespace_exec.php

Exec.: http://localhost/dev/docsPHPOO/10_exemplo_namespace_exec.php

Autor: Patrick Silva Ferraz

Informações adicionais: Comparando objetos

- **Comparando objetos**
  - Evita “colisões de nomes” (quando classes, funções, variáveis ou constantes com o mesmo nome);
  - (==): São iguais se possuem os memos atributos e volores, e são instâncias da mesma classe.
  - (===): São idênticas, se e somente se, referirem a mesma instância da mesma classe.

Autor: Patrick Silva Ferraz

Informações adicionais: Constantes Mágicas

- **Constantes Mágicas**
  - __LINE__ - Número da linha corrente;
  - __FILE__ - Caminho completo e nome do arquivo;
  - __DIR__ - Diretório do arquivo;
  - __FUNCTION__ - Nome da função;
  - __CLASS__ - Nome da classe;
  - __TRAIT__ - Nome do trait;
  - __METHOD__ - Nome do método da classe;
  - __NAMESPACE__ - Nome do namespace corrente;
  - NomeDaClasse::class – Nome completo da classe;

Infor.: http://us3.php.net/manual/pt_BR/language.constants.predefined.php

Referências

- http://php.net/manual/pt_BR/language.oop5.php
- http://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264
- https://code.tutsplus.com/pt/tutorials/object-oriented-php-for-beginners--net-12762
- http://blog.thiagobelem.net/usando-namespaces-no-php