SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

2016.2

Introdução a Programação Orientada a Objetos Utilizando PHP PARTE 04

Docente: Gil Jader – gil.jader@gmail.com

1. Intercepções (Métodos Mágicos)

O PHP5 introduziu o conceito de interceptação em operações realizadas por objetos por meio de métodos inicializados por __ (*undeline* duplo)

O PHP reserva todas as funções com nomes iniciadas com __ como mágicas. É recomendado que não se utilize funções com nomes com __ no PHP.

1. Intercepções (Métodos Mágicos)

- **Método __set()**

O método __set() intercepta a atribuição de valores a propriedades do objeto. Sempre que for **atribuído** um valor a uma propriedade do objeto, automaticamente esta atribuição passa pelo método __set(), o qual recebe o **nome da propriedade** e o **valor** a ser atribuído, pode atribuí-lo ou não.

1. Intercepções (Métodos Mágicos)

- **Método __set()**

```php
class GetSet
{
    private $nome;
    private $sobrenome;

    function __set($atrib, $value)
    {
        $this->$atrib = $value;
    }
```

1. Intercepções (Métodos Mágicos)

- **Método __get()**

O método __get() intercepta requisições de propriedades. Sempre que for **requisitada** uma propriedade, automaticamente essa requisição passará pelo método __get(), o qual recebe o nome da propriedade requisitada, podendo retorná-la ou não.

1. Intercepções (Métodos Mágicos)

- **Método __get()**

```php
class GetSet
{
    private $nome;
    private $sobrenome;

    function __set($atrib, $value)
    {
        $this->$atrib = $value;
    }

    function __get($atrib)
    {
        return $this->$atrib;
    }
}
```

1. Intercepções (Métodos Mágicos)

- **Exemplo de utilização do métodos __set e __get()**

```php
$pessoa = new GetSet();
$pessoa->nome = 'Gil';
$pessoa->sobrenome = 'Jader';

echo 'Nome: ' . $pessoa->nome . '<br>';
echo 'Sobrenome: ' . $pessoa->sobrenome;
```

1. Intercepções (Métodos Mágicos)

- **Método __call()**

O método __call() intercepta a **chamada de métodos**. Sempre que for executado um método que não existir no objeto, a execução será direcionada para ele, que **recebe dois parâmetros**, o **nome do método** requisitado e o **parâmetro recebido**, podendo decidir o procedimento a realizar.

1. Intercepções (Métodos Mágicos)

- **Método __call()**

```php
<?php

class ClasseComTodosOsMetodosDoMundo
{
    public function __call($m, $a)
    {
        echo "Método invocado: $m\n";
    }
}

$o = new ClasseComTodosOsMetodosDoMundo();
$o->umMetodoQualquer();
$o->umMetodoAssimAssado();

?>
```

Resultado:

```
Método invocado: umMetodoQualquer
Método invocado: umMetodoAssimAssado
```

1. Intercepções (Métodos Mágicos)

- **Método __toString()**

Quando imprimimos objetos na tela, por meio de comandos como **echo** e **print**, o PHP exibe no console o identificador interno do objeto, por exemplo:

Object id #1

Object id #2

Para alterar esse comportamento, podemos definir o método **__tostring()** para cada classe. Caso o método **__tostring()** exista, no momento em que mandamos exibir um objeto no console, o PHP **irá imprimir o retorno** dessa função.

Programação Web com Software Livre – 2016.2

Exemplo:

```php
<?php
// Declare a simple class
class TestClass
{
    public $foo;

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

    public function __toString()
    {
        return $this->foo;
    }
}

$class = new TestClass('Hello');
echo $class;
?>
```

1. Intercepções (Métodos Mágicos)

- **Método __clone()**
  - O comportamento padrão do PHP quando atribuímos um objeto a outro é criar uma **referência entre objetos**. Dessa forma, teremos duas variáveis apontando para a mesma região da memória.
  - Como proceder quando precisamos **duplicar** um objeto na memória?
  - Utilizando o método __clone, responsável por definir o comportamento da ação de clonagem, atuando diretamente nas propriedades do objeto resultante dessa ação.

```php
<?php
class Pessoa
{
    // Propriedades da classe
    public $nome;
    public $sobrenome;
    public $idade;

    // Construtor - Define os valores das proprieades
    public function __construct ( $nome = null, $sobrenome = null, $idade = 0 ) {
        $this->nome      = $nome;
        $this->sobrenome = $sobrenome;
        $this->idade     = (int) $idade;
    }

    // Método para exibir
    public function exibir () {
        echo 'Nome: ';
        echo $this->nome . '<br>';
        echo 'Sobrenome: ';
        echo $this->sobrenome . '<br>';
        echo 'Idade: ';
        echo $this->idade . '<br><br>';
    }

    // Método executado ao criar um clone
    public function __clone () {
        // Se for um clone, adiciona a palavra "Clonado" no valor
        $this->nome      = $this->nome . ' (Clonado)';
        $this->sobrenome = $this->sobrenome . ' (Clonado)';
        $this->idade     = $this->idade . ' (Clonado)';
    }
}
```

1. Intercepções (Métodos Mágicos)

- **Método __clone()**

Exemplo de utilização da classe anterior

```php
// Instância da classe Pessoa
$pessoa = new Pessoa('Luiz', 'Miranda Figueiredo', 27);

// Aqui está o clone
$pessoa2 = clone $pessoa;
$pessoa2->exibir();
```

1. Intercepções (Métodos Mágicos)

- **Método __autoload()**

  - Sempre que se instancia um objeto em PHP, **é necessário ter a declaração da classe na memória** e podemos fazer isso, por exemplo, utilizando o **include_once**. Podemos realizar tal operação no início da aplicação introduzindo todas as classes que poderão ser necessárias durante a execução da aplicação.

  - Para simplificar tal procedimento, o PHP introduziu a função **__autoload()**, ou "carga automática". Com ela, a carga da classe é **realizada de forma dinâmica sempre que um objeto for instanciado**.

  - Esta função recebe o nome da classe que será instanciada e introduz a classe na memória.

1. Intercepções (Métodos Mágicos)

- **Método __autoload()**

Exemplo: autoload.php

```php
<?php

# função de carga automática
function __autoload($classe)
{
    # busca no diretório de classes...
    include_once "classes/{$classe}.class.php";
}

# instanciando um novo Produto
$bolo = new Produto(500, 'Bolo de Fubá', 4, 4.12);
echo 'Código: ' . $bolo->Codigo . "<br>";
echo 'Descrição: ' . $bolo->Descricao . "<br>";

?>
```

2.Interfaces

- A programação orientada a objetos baseia-se fortemente na interação de classes e objetos. Na etapa de projeto do sistema, podemos definir **conjunto de métodos** que determinadas classes do nosso sistema **deverão implementar incondicionalmente**.

- Uma interface em orientação a objetos **não é considerada uma classe** e sim uma **entidade**, **não possui implementação**, **apenas assinatura**, ou seja, apenas a definição de seus métodos sem o corpo. Não há como fazer instância de uma interface e portanto funcionam como um tipo de “contrato”, onde são definidos os **métodos que as classes que implementem essa interface são obrigado a especificar**.

2.Interfaces

Exemplo:

```php
<?php

// Declara a interface 'iTemplate'
interface iTemplate
{
    public function setVariable($name, $var);
    public function getHtml($template);
}
```

O que acontece aqui?

```php
// Isso NÃO funcionará
// Fatal error: Class BadTemplate contains 1 abstract methods
// and must therefore be declared abstract (iTemplate::getHtml)
class BadTemplate implements iTemplate
{
    private $vars = array();

    public function setVariable($name, $var)
    {
        $this->vars[$name] = $var;
    }
}
?>
```

2.Interfaces

Exemplo:

```php
// Isso funcionará
class Template implements iTemplate
{
    private $vars = array();

    public function setVariable($name, $var)
    {
        $this->vars[$name] = $var;
    }

    public function getHtml($template)
    {
        foreach($this->vars as $name => $value) {
            $template = str_replace('{' . $name . '}', $value, $template);
        }

        return $template;
    }
}
```

REFERÊNCIAS

- Dall'Oglio, Pablo. Php 5 - PHP: Programando com Orientação a Objetos. 3ª Edição, São Paulo, Novatec, 2015.
- WALLACE, Soares. Php 5 - Conceitos, Programação e Integração com Banco de Dados. 7ª Edição, São Paulo, Erica, 2013.
- http://www.php.net/manual/pt_BR/index.php