ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

Programação Orientada a Objetos

Profº Ricardo Aragão - raragao@raragao.eng.br
Zend Certified Engineer PHP 5

Declaração de classes

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```php
class SimpleClass
{
    // declaração de membro
    public $var = 'um valor padrão';

    // declaração de método
    public function displayVar() {
        echo $this->var;
    }
}
```

Instâncias (objetos)

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```
instance = new SimpleClass();
$assigned   =  $instance;
$assigned->var = 'teste';
print $instance->var;

$reference  =& $instance;

$instance->var = '$assigned terá esse valor';
$instance = null; // $instance e $reference tornam-se nulos

var_dump($instance);
var_dump($reference);
var_dump($assigned);
```

Carregamento automático

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```php
function __autoload($class_name) {
    require_once $class_name . '.php';
}

$obj  = new MyClass1();
$obj2 = new MyClass2();
```

Herança

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```
class a {
    function metodoA(){
        echo "a::metodoA chamado";
    }
    function metodoB(){
        echo "a::metodoB chamado";
    }
}

class b extends a {
    function metodoA() {
        echo "b::metodoA chamado";
    }
}
```

Herança

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```
$a = new a();

$b = new b();


$a->metodoA();

$b->metodoA();

$b->metodoB();
```

Referencia ao próprio objeto

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```php
class MinhaClasse {
    function exibir($dado) {
        echo "O valor é $dado";
    }
    function chamarExibir($dado) {
        // Call myFunction()
        $this->exibir($dado);
    }
}
$obj = new MinhaClasse();
$obj->chamarExibir(123);
```

Construtor

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```
class ClasseBase {
    function __construct() {
        print "No construtor da ClasseBase\n";
    }
}
class SubClasse extends ClasseBase {
    function __construct() {
        parent::__construct();
        print "No construtor da SubClasse\n";
    }
}
$obj = new ClasseBase();
$obj = new SubClasse();
```

Destrutor

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```
class MinhaClasseDestruivel {
    function __construct() {
        print "No construtor\n";
        $this->name = "MinhaClasseDestruivel";
    }

    function __destruct() {
        print "Destruindo " . $this->name . "\n";
    }
}

$obj = new MinhaClasseDestruivel();
```

Visibilidade

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

public	o recurso pode ser acessado de qualquer escopo
protected	o recurso só pode ser acessado de dentro da classe onde está definido ou de dentro de classes filhas
private	o recurso só pode ser acessado de dentro da classe onde está definido
final	o recurso pode ser acesso de qualquer escopo mas não pode ser

Visibilidade de atributos

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```php
class MinhaClasse {
    public $publica = 'Public';
    protected $protegida = 'Protected';
    private $privada = 'Private';
    function imprimeAlo() {
        echo $this->publica;
        echo $this->protegida;
        echo $this->privada;
    }
}
$obj = new MinhaClasse();
echo $obj->publica; // Funciona
echo $obj->protegida; // Erro Fatal
echo $obj->privada; // Erro Fatal
$obj->imprimeAlo(); // Mostra Public, Protected e Private
```

Visibilidade de atributos

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```
class MinhaClasse2 extends MinhaClasse {

    // Nós podemos redeclarar as propriedades públicas e
  protegidas mas não as privadas

    protected $protegida = 'Protected2';

    function imprimeAlo() {

        echo $this->publica;

        echo $this->protegida;

        echo $this->privada;

    }

}

$obj2 = new MinhaClasse2();

echo $obj2->publica; // Works

echo $obj2->privada; // Undefined

echo $obj2->protegida; // Fatal Error

$obj2->imprimeAlo(); // Mostra Public, Protected2, not Private
```

Visibilidade de métodos

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```php
class MinhaClasse {
    public function __construct() { }
    public function MeuPublico() { }
    protected function MeuProtegido() { }
    private function MeuPrivado() { }
    function Foo() {
        $this->MeuPublico();
        $this->MeuProtegido();
        $this->MeuPrivado();
    }
}
$minhaclasse = new MinhaClasse;
$minhaclasse->MeuPublico(); // Funciona
$minhaclasse->MeuProtegido(); // Erro Fatal
$minhaclasse->MeuPrivado(); // Erro Fatal
$minhaclasse->Foo(); // Public, Protected e Private funcionam
```

Visibilidade de métodos

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```
class MinhaClasse2 extends MinhaClasse {

    // Esse é public

    function Foo2() {

        $this->MeuPublico();

        $this->MeuProtegido();

        $this->MeuPrivado(); // Erro Fatal

    }

}


$minhaclasse2 = new MinhaClasse2;

$minhaclasse2->MeuPublico(); // Funciona

$minhaclasse2->Foo2(); // Public e Protected funcionam, Private
  não
```

Operador ::, constante e static

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```php
class MinhaClasse {
    const VALOR_CONST = 'Um valor constante';
}
echo MinhaClasse::VALOR_CONST;

class OutraClasse extends MinhaClasse {
  public static $meu_estatico = 'variável estática';
  public static function doisPontosDuplo() {
      echo parent::VALOR_CONST . "\n";
      echo self::$meu_estatico . "\n";
  }
}
OutraClasse::doisPontosDuplo();
```

Sobrecarga

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```
class OutraClasse extends MinhaClasse {

  /* Sobrecarrega a definição do pai */
  public function minhaFuncao() {

    /* Mas ainda chama a função pai */
    parent::minhaFuncao();
    echo "OutraClasse::minhaFuncao()\n";
  }
}

$classe = new OutraClasse();
$classe->minhaFuncao();
```

Classes e métodos abstratos

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```php
abstract class ClasseAbstrata
{
    abstract protected function pegarValor();
    abstract protected function valorComPrefixo( $prefixo );
    public function imprimir() {
        print $this->pegarValor();
    }
}

class ClasseConcreta1 extends ClasseAbstrata {
    protected function pegarValor() {
        return "ClasseConcreta1";
    }
    public function valorComPrefixo( $prefixo ) {
        return "{$prefixo}ClasseConcreta1";
    }
}
```

Interface

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

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

Polimorfismo

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```
class Cat {
    function miau() {
        print "miau";
    }
}
class Dog {
    function wuff() {
        print "wuff";
    }
}
function printTheRightSound($obj) {
    if ($obj instanceof Cat) {
        $obj->miau();
    } else if ($obj instanceof Dog) {
        $obj->wuff();
    }
}
printTheRightSound(new Cat());
printTheRightSound(new Dog());
```

Iteração interna

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```
class MyClass {
    public $var1 = 'value 1';
    public $var2 = 'value 2';
    public $var3 = 'value 3';

    protected $protected = 'protected var';
    private   $private   = 'private var';

    function iterateVisible() {
        echo "MyClass::iterateVisible:\n";
        foreach($this as $key => $value) {
            print "$key => $value\n";
        }
    }
}
```

Iteração externa

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```php
$class = new MyClass();


foreach($class as $key => $value) {
    print "$key => $value\n";
}
echo "\n";
```

Iteração usando Iterator

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```
class MyIterator implements Iterator {
  private $var = array();
  public function __construct($array)
  {
    if (is_array($array) ) {
      $this->var = $array;
    }
  }
  public function rewind() {
    reset($this->var);
  }
  public function current() {
    $var = current($this->var);
    return $var;
  }
  public function key() {
    $var = key($this->var);
    return $var;
  }
  public function next() {
    $var = next($this->var);
    return $var;
  }
  public function valid() {
    $var = $this->current() !== false;
    return $var;
  }
}

$values = array(1,2,3);
$it = new MyIterator($values);

foreach ($it as $a => $b) {
  print "$a: $b\n";
}
```

Iteração usando IteratorAggregate

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```php
class MyCollection implements IteratorAggregate {
  private $items = array();
  private $count = 0;
  /* Definião requirida da interface IteratorAggregate */
  public function getIterator() {
    return new MyIterator($this->items);
  }
  public function add($value) {
    $this->items[$this->count++] = $value;
  }
}

$coll = new MyCollection();
$coll->add('value 1');
$coll->add('value 2');
$coll->add('value 3');
foreach ($coll as $key => $val) {
  echo "key/value: [$key -> $val]\n\n";
}
```

Exceção

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```php
try {
  $error = 'Sempre dispara esse erro';
  throw new Exception($error);

  // Código que segue uma exceção não é executado
  echo 'Nunca é executado';
}
catch (Exception $e) {
  echo "Exceção pega: ",  $e, "\n";
}
```

Classe Exception nativa

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```
class Exception {
  protected $message = 'Unknown exception'; // Mensagem da exceção
  protected $code = 0; // Código da exceção definido pelo usuário
  protected $file; // Arquivo gerador da exceção
  protected $line; // Linha geradora da exceção
  function __construct(string $message=NULL, int code=0);
  final function getMessage();           // Mensagem da exceção
  final function getCode();              // Código da exceção
  final function getFile();              // Arquivo gerador
  final function getTrace();             // um array com o backtrace()
  final function getTraceAsString();     // String formatada do trace
  /* Sobrecarregável */
  function _toString();  // String formatada para ser mostrada

}
```

Indução de Tipo

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```php
class MinhaClasse
{
    public function teste(OutraClasse $outraclasse) {
        echo $outraclasse->var;
    }
    public function testa_array(array $array_de_entrada) {
        print_r($array_de_entrada);
    }
}

class OutraClasse {
    public $var = 'Alô Mundo';
}
```

Métodos mágicos

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

- Os nomes de funções __construct, __destruct (veja Construtores e Destrutores), __call, __get, __set, __isset, __unset (veja Sobrecarga), __sleep, __wakeup, __toString, __set_state, __clone and __autoload são mágicos nas classes do PHP. Você não pode ter funções com esses nomes em nenhuma de suas classes a não ser que queria que a funcionalidade mágica associada com eles.

Clonando objetos

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

```php
$copia_do_objeto = clone $objeto;
```

- Quando um objeto é clonado, PHP 5 fará uma cópia superficial de todas as propriedades do objeto. Qualquer propriedade que seja referência a outra variável, permanecerá referência. Se um método __clone() for definido, então este será chamado, permitindo definir qualquer alteração necessária nas propriedades.

Comparação de objetos

ESCOLA DE FORMAÇÃO DE TRABALHADORES EM INFORMÁTICA

- No PHP 5, comparação de objetos é mais complicada que no PHP 4 e mais de acordo com o que é esperado de uma Linguagem Orientada a Objetos (não que PHP 5 seja uma delas).
- Quando usar o operador de comparação (==), variáveis objeto são comparadas de maneira simples, nominalmente: Duas instâncias de objetos são iguais se tem os mesmos atributos e valores, e são instâncias da mesma classe.
- Por outro lado, quando usando o operador de identidade (===), variáveis objetos são identicas se e somente se elas se referem a mesma instância da mesma classe.