Programação Orientada a Objetos com PHP

Fernando Lozano

http://www.lozano.eti.br

Consultor Independente

Prof. Faculdade Metodista Bennett

Prof. Instituto de Tecnologia ORT

Sobre o Autor

- Red Hat Certified Engineer
  LPI Certified Professional Level I
  Sair GNU/Linux Certified Professional
- IBM Certified Network Engineer
  IBM Certified DB2 Administrator & Developer
- Microsoft Certified Systems Engineer
  Microsoft Certified Solutions Developer
- Webmaster da Free Software Foundation
- Colaborador da Revista do Linux e da Revista PC Master
- Autor do Livro "*Java em GNU/Linux*"
  ed. Alta Books www.altabooks.com.br

Agenda

- Revisão de Orientação a Objetos
- Definindo classes em PHP
- Utilizando classes em PHP
- PEAR, a biblioteca de classes do PHP
- Classes para Bancos de Dados
- PHP e GTK+
- PHP e Java
- XP e PHP: PhpUnit

Revisão de Orientação a Objetos

Conceitos Essenciais de OO

- Objeto
  Representa uma coisa física, tangível, uma idéia ou conceito. Possui um estado (o que ele sabe) e um comportamento (o que ele é capaz de fazer, como ele reage a estímulos externos).

Conceitos Essenciais de OO

- Classe
  É um "molde" para a criação de objetos, fornecendo o seu comportamento padrão e a definição de todos os seus estados possíveis.
  Ex: Classe Correntista
- Instância
  É uma ocorrência particular, identificada, de um objeto de uma determinada classe, com seu estado particular, independente de outras instâncias da mesma classe.
  Ex: o objeto Correntista "Fernando Lozano"

Conceitos Essenciais de OO

- Encapsulamento
  Um objeto contém todas as informações (variáveis) e toda a inteligênica (código) de que necessita para realizar suas atribuições. Ele deve ser tanto quanto possível auto-contido, independente de informações ou código que não façam parte dele mesmo.
- Ocultamento de Informações
  Deve ser possível utilizar um objeto apenas pelo conhecimento da sua estrutura externa (isto é, sua interface). Mudanças na estrutura interna de um objeto (isto é, sua implementação) não devem afetar aos usuários do objeto.

Conceitos Essenciais de OO

- Polimorfismo
  A mesma mensagem, quando enviada para objetos de classes diferentes, executa código particular da classe, mesmo que quem enviou a mensagem não tenha conhecimento do tipo específico de objeto sendo referenciado.
- Herança ou Especialização
  Uma nova classe pode ser definida em termos de uma classe pai, herdando o seu comportamento. A nova classe especializa a classe pai, defindo apenas onde o seu comportamento deve ser diferente.

Conceitos Essenciais de OO

- Agregação e Composição
  Objetos podem conter outros objetos como partes constituintes, imitando o mundo real onde objetos são construídos em função de outros objetos. Podemos ou não expor as partes constituintes como parte da interface de um objeto.

Classes e Objetos em PHP

Definindo Classes em PHP

- A palavra-chave class indica uma declaração de classe, delimitada por chaves.
- Dentro da classe podemos definir atributos (variáveis) e métodos (funções) que formam o estado e o comportamento do objeto.
- Um método com o mesmo nome da classe é o construtor do objeto, sendo executado sempre que uma instância for criada.
- A classe deve utilizar a variável $this para referenciar seus próprios métodos e atributos

Uma Classe Simples

- class ContaCorrente
  {
  var $saldo;

  function ContaCorrente ($valor) {
  $this->saldo = $valor;
  }
  function saque ($valor) {
  if ($this->saldo >= $valor)
  $this->saldo -= $valor;
  }
  function deposito ($valor) {
  $this->saldo += $valor;
  }
  }

Utilizando uma Classe

- A definição da classe deve estar disponível no script ou página PHP que utiliza a classe (comandos include ou require).
- Um objeto da classe deve ser instanciado pelo operador new.
- O operador -> permite referenciarmos atributos e métodos do objeto

Utilizando a Classe

- ```
  <?php
      include "ContaCorrente.php";

      $conta = new ContaCorrente (1000.0);
      echo ("Saldo inicial: {$conta->saldo}<BR>");
      $conta->saque (150.0);
      echo ("Novo saldo: {$conta->saldo}<BR>");
  ?>
  ```
  ```

Herança Em PHP

- Uma classe pode extender (extends) outra classe qualquer
- Todos os atributos e métodos estão disponíveis imediatamente, pela variável $this
- O construtor da superclasse deve ser chamado explicitamente pelo construtor da subclasse
- A definição da subclasse deve incluir a definição da superclasse; utilize o comando include_once (ou então require_once) para evitar problemas

Uma Subclasse em PHP

- Include_once "ContaCorrente.php";

```
class ContaEspecial extends ContaCorrente
{
        var $limite;

        function ContaEspecial ($valor, $limite) {
                $this->ContaCorrente ($valor);
                $super ($valor);
        }
        function saque ($valor) {
                if ($this->saldo + $this->limite >= $valor)
                        $saldo -= $valor;
        }
}
```

Utilizando a Subclasse

- ```
  <?php
      include_once "ContaCorrente.php";
      include_once "ContaEspecial.php";

      $conta = new ContaCorrente (1000.0);
      echo ("Saldo inicial: {$conta->saldo}<BR>");
      $conta->saque (1400.0);
      echo ("Novo saldo: {$conta->saldo}<BR>");

      $contaesp = new ContaEspecial (1000.0, 500.0);
      echo ("Saldo inicial: {$contaesp->saldo}<BR>");
      $contaesp->saque (1400.0);
      echo ("Novo saldo: {$contaesp->saldo}<BR>");
  ?>
  ```
  ```

Métodos de Classe

- Qualquer método pode ser chamado como um método de classe (isto é, sem que haja uma referência a uma instância da classe) utilizando o operador ::
- O método de classe irá falhar caso faça referência à variável $this
- Não existem atributos de classe

Um Método de Classe

- class CaixaEletronico extends CaixaBancario
  {
  // ...
  function limiteSaques () {
  if ($agencia->turno () == NOTURNO)
  return 100.00;
  else
  return 0;
  }
  // ...
  }

Utilizando um Método de Classe

-
```
<?php
    // $conta é a ContaEspecial sendo manipulada
    // $valor é a quantia sendo sacada

    if (CaixaEletronico::limiteSaques () >= $valor)
        $conta->saque ($valor);
?>
```

Recursos OO Ausentes do PHP

- Métodos destrutores
- Herança múltipla
- Restrições de visibilidade
  (public, protected, private)
- Classes abstratas ou interfaces
- Casts
  (não são necessários, visto que PHP não é tipado)
- Excessões

Consulte o Manual sobre...

- Referências
  O PHP passa parâmetros sempre por valor, o que pode gerar resultados inesperados quando se utiliza um objeto como argumento para uma função - ela recebe uma cópia do objeto, e não o próprio objeto
- Serialização
  O PHP é capaz de salvar qualquer tipo de objeto em uma string. Útil para seções, arquivos, persistência, ...
- Identificação de tipos, Reflexão e Introspecção
  (Tópico: funções para Classes e Objetos)

Exemplo: Listando os Métodos de um Objeto

-
```
<?php
    $conta = new ContaCorrente (1000.0);

    $metodos = get_class_methods (get_class ($conta));

    foreach ($metodos as $nome_metodo) {
       echo "$nome_metodo<BR>";
    }
?>
```

Porque Usar Classes em PHP

- Utilizar uma sintaxe mais clara e limpa para operações complexas, que requeiram uma grande quantidade de argumentos
- Tornar programas complexos mais facilmente gerenciáveis e manuteníveis pelo agrupamento de variáveis e código inter-relacionados
- Fomentar a reutilização de código, criando componentes genéricos
- Permitir a substituição de componentes de um sistema, em estilo plug-ins

A Biblioteca de Classes PEAR

A Biblioteca PEAR

- A construção do PEAR se iniciou no PHP4 de modo a fornecer uma biblioteca padrão de classes para as operações mais comuns em aplicativos PHP, inspirada no CPAN para Perl
- Antes do PEAR haviam várias bibliotecas de classes populares, como a PHPlib, mas a distribuição padrão do PHP fornecia apenas módulos procedurais (bibliotecas de funções)

A Biblioteca PEAR

- Ela é mantida como um projeto autônomo dentro do PHP, com documentação e etc à parte, mas é fornecida em cada novo release do PHP
- Especifica convenções que devem ser seguidas por quaisquer classes reutilizáveis no PHP
- Fornece uma emulação de destrutores para garantir que não hajam "leaks" de recursos
- Fornece um mecanismo padrão de tratamento de erros para as classes da biblioteca

Componentes Fornecidos Pelo PEAR

- Acesso a bancos de dados
- E-mail internet
- Opções de linha de comando
- Conexões TCP e HTTP
- Autenticação
- Templates HTML
- XML
- Documentação e Testes automatizados

Destrutores com PEAR

- Toda classe deve ser derivada da classe PEAR e chamar o construtor da superclasse
- Os objetos devem ser instanciados como referências (&new)
- O método _NomeDaClasse de cada objeto é invocado automaticamente ao fim do script / página PHP

Exemplo de Destrutor

- class CaixaEletronico extends PEAR
  {
  // ... atributos do caixa
  function CaixaEletronico () {
  $this->PEAR ();
  $log = fopen ($nome, "a");
  // ... outras inicializações
  }
  // ... utiliza $log para registrar todas as operações realizadas
  // ... pelo caixa com fins de auditoria
  function _CaixaEletronico () {
  if (is_resource ($log))
  fclose ($log);
  // ... outras tarefas para "limpar a casa"
  }
  }

Tratamento de Erros com PEAR

- Qualquer método que possa gerar erros deve retornar uma subclasse de PEAR_Error
- O usuário da classe pode então testar se recebeu um objeto de erro como retorno
- Cada objeto derivado de PEAR pode definir que erros obrigam o programa a ser abortado (sendo executada a função de tratamento de erros definida para o script / página, em geral para a exibição de uma página de erro personalizada)

Método Que Gera Erros

- class ContaCorrente extends PEAR
  {
  // ...
  function saque ($valor) {
  if ($this->saldo <= $valor)
  return $this->raiseError ("Saldo insuficiente",
  ERR_VALOR);
  else
  $this->saldo -= $valor;
  }
  // ...
  }

Usuário do Método

- ```
  <?php
      $conta = &new ContaCorrente (1000.0);

      $err = $conta->saque (1500.0);
      if (PEAR::isError ($err)) {
          echo ("Não pude completar o saque:");
          echo ($err->getMessage ());
      }
  ?>
  ```
  ```

Classes Para Bancos de Dados

Classes DB do PEAR

- Fornecem uma API padrão para todos os bancos de dados suportados pelo PHP: não é mais preciso reescrever o código quando se muda de banco
- Necessitam dos módulos específicos para cada banco compilados / carregados no PHP, ou seja, o acesso ainda é nativo
- A conexão a um banco é feita por um "método fábrica" que fornece um objeto especializado para o banco sendo utilizado no momento

Classes DB do PEAR

- DB

  Representa uma conexão a um banco de dados qualquer, fornecendo métodos para execução de comandos SQL e geração de sequências.

- DB_Result

  Representa o resultado de um comando SELECT ou procedimento armazenado (ou seja, representa um cursor). Fornece métodos para a iteração pelas linhas / registros retornados e obtenção de meta-dados sobre o resultado.

Acesso a BD Sem o PEAR

```
<?php
    $con = mysql_connect ("localhost", "teste", "senha");
    mysql_select_db ("banco");
    $sql = "SELECT nome, e_mail " .
        "FROM funcionario ORDER BY nome";
    $rs = mysql_query ($sql, $con);
    echo ("<TABLE border>");
    echo ("<TR><TD>Nome</TD><TD>E-mail</TD></TD>");
    while ($row = mysql_fetch_row ($rs)) {
        echo ("<TR><TD>{$row[0]}</TD>");
        echo ("<TD>{$row[1]}</TD></TR>");
    }
    echo ("</TABLE>");
    mysql_free_result ($rs);
    mysql_close ($con);
?>
```

Acesso a BD com o PEAR

```
<?php
    include "DB.php";
    $con = DB::connect ("mysql://agenda:teste@localhost/banco");
    $sql = "SELECT nome, e_mail " .
        "FROM funcionario ORDER BY nome";
    $rs = $con->query ($sql);
    echo ("<TABLE border>");
    echo ("<TR><TD>Nome</TD><TD>E-mail</TD></TD>");
    while ($row = $rs->fetchRow ()) {
        echo ("<TR><TD>{$row[0]}</TD>");
        echo ("<TD>{$row[1]}</TD></TR>");
    }
    echo ("</TABLE>");
    $rs->free ();
    $con->disconnect ();
?>
```

Uma URL (ou DSN) PEAR DB

- protocolo://login:senha@host:porta/banco
- Protocolo
  Identifica o banco. Ex: mysql, pgsq, oci8, mssql, ...
- Login:Senha
  Nome do usuário e senha para conexão ao banco
- Host:Porta
  Nome ou endereço IP do servidor, opcionalmente seguido da porta TCP
- Banco
  Apenas para servidores que gerenciam múltiplos bancos

Tratamento de Erros do Banco

- ```
  $con = DB::connect ($dsn);
  if (DB::isError ($con)) {
        // ...
  }

  $rs = $con->query ($sql);
  if (DB::isError ($rs)) {
        // ...
  }

  while ($row = $rs->fetchRow ()) {
        if (DB::isError ($row)) {
              // ...
        }
  }
  ```
  ```

PHP e GTK+

O Toolkit GTK+

- Criado originalmente para o desenvolvimento do GIMP (GNU Image Manipulation Program)
- Design OO implementado em C procedural, o que torna fácil suporta-lo em outras linguagens
- Adotado por inúmeras aplicações Linux/Unix
- Utilizado como base para o desktop Gnome
- Utilizado em aplicativos multiplataforma como o Mozilla e o StarOffice
- Suporte a Windows, Mac e BeOS

Porque Utilizar GTK+ com o PHP?

- InterfacesWeb não são ideais para todos os tipos de aplicações
- Dada a grande quantidade de know-how e código pronto que uma empresa acumula no desenvolvimento em PHP, seria interessante reutilizar tudo isto quando há necessidade de uma interface gráfica tradicional
- O GTK+ mantém o caráter multiplataforma dos programas PHP com interface web

GTK+ versus Web

- A mesma compilação do PHP não pode ser utilizada para aplicações web e para aplicações GUI. Deve ser gerado um executável "CGI" do PHP com o módulo GTK+ habilitado, e este executável deve ser utilizado como um shell para execução de scripts.
- A geração de um PHP "CGI" já é uma tarefa comum em web sites de produção, para que certas páginas possam ser geradas periodicamente em retaguarda, em vez de interativamente a cada visita do usuário; ou para a codificação de rotinas de back office em PHP.

Conceitos do GTK+

- Widgets
  Qualquer tipo de componente visual
- Conteiner
  Componentes visuais que podem conter outros componentes
- Sinais
  Eventos gerados pelo usuário
- Callbacks
  Rotinas de tratamento de eventos no programa

Hello, Word PHP com GTK+

-
```
<?php
    $win = &new GtkWindow ();
    $win->set_border_width (8);
    $win->connect_object ("destroy", array ("gtk", "main_quit"));

    $msg = &new GtkLabel ("Oi do PHP com GTK+!");
    $win->add ($msg);
    $win->show_all ();

    Gtk::main ();
?>
```

Explicando o Programa

- Gtk::main é o loop de eventos do GTK+. Nada acontece enquanto este loop não é inicializado
- connect_object permite que métodos de alguma classe sejam associados a um evento. No caso associamos o evento "destroy" ao método gtk::main_quit, que encerra o loop de eventos do GTK+
- Add insere um widget dentro de um conteiner
- Todos os widgets são inicialmente invisíveis, por isso a chamada a show_all

Boxes e Tables

- No GTK+ não posicionamos os componentes pelas suas coordenadas em pixels, pois isto inviabilizaria aplicações multiplataforma (além do suporte a temas do GTK+).
- Em vez disso, utilizamos conteiners Box e Table para posicionar os componentes uns em relação aos outros.
  (Programadores Java irão notar a semelhança com os layout managers do AWT e Swing)

Contador de Clicks

```
$win = &new GtkWindow ();
$win->set_border_width (8);
$win->connect_object ("destroy", array ("gtk", "main_quit"));

$box = &new GtkVBox (false, 4);
$win->add ($box);

$msg = &new GtkLabel ("Contador de clicks");
$box->pack_start ($msg, true, true);

$ok = &new GtkButton ("Mais Um!");
$ok->connect ("clicked", "incrementa");
$box->pack_start ($ok, false, false);

$win->show_all ();
Gtk::main ();
```

O Evento Clicked

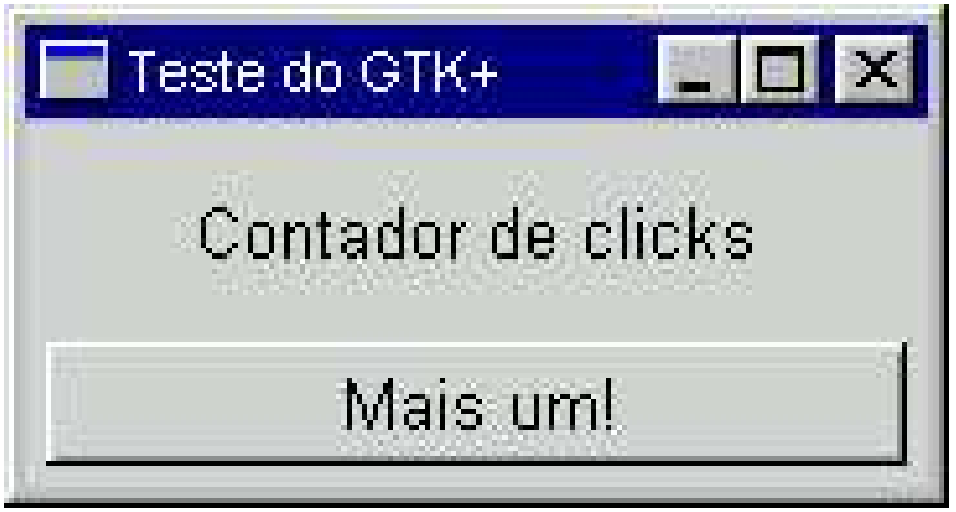
-

```
$contador = 0;

function incrementa () {
        global $contador;
        global $msg;

        $contador++;
        $msg->set_text ("Contei $contador clicks.");
}
```

GUI Multiplataforma



PHP e Glade

- O Glade é o construtor de janelas padrão do Gnome, mas ele suporta apenas C, Perl e Ada
- A libGlade permie gerar janelas dinamicamente à partir do arquivo XML gerado pelo Glade
- O PHP-GTK+ suporta a libGlade, de modo que podemos utilizar o Glade para desenhar interfaces gráficas de aplicações PHP
- O wGlade é o porte do Glade para Windows

PHP e Java

Porque PHP com Java

- Java vem se tornando a "linguagem de sistemas" preferencial do mercado, ocupando o lugar que antes era de C e C++
- Grandes sistemas web, operando em três camadas (computação distribuída) utilizam na maioria das vezes o Enterprise Java
- Portanto, há uma grande quantidade de código pronto em Java em muitas grandes empresas
- Mas Java é uma linguagem complexa. É mais fácil encontrar desenvolvedores comn experiência em PHP

Java Dentro do PHP

- O PHP é capaz de interagir com uma máquina virtual "embutida" via JNI e acessar objetos Java como se fossem objetos PHP
- O PHP cuida automaticamente da conversão de tipos de dados do Java para o PHP e vice-versa
- O PHP pode ser compilado para rodar como um servlet em servidores de aplicações Java
- O CLASSPATH é fornecido no arquivo php.ini

PHP e Java

- A classe "Java" do PHP instancia a classe Java cujo nome é fornecido no construtor
- As funções java_last_exception_get e java_last_exception_clear permitem ao código PHP tratar excessões geradas pelo Java
- O código PHP pode então acessar objetos Java remotos via RMI ou mesmo EJBs
- Se compilarmos o PHP como "CGI" podemos construir interfaces gráficas utilizando Swing ou AWT
  (mas elas serão tão pesadas quanto seriam em Java)

Exemplo: Informações Sobre a Máquina Virtual Java

- ```
  <?php
      $system = new Java ("java.lang.System");

      echo ("Java version=" . $system->getProperty ("java.version"));
      echo ("<BR>");;
      echo ("Java vendor=" . $system->getProperty ("java.vendor")):
      echo ("<BR>");;
      echo ("OS=" . $system->getProperty ("os.name") . " ");
      echo ($system->getProperty ("os.version"));
      echo ("<BR>");;
      echo ("CPU=" . $system->getProperty ("os.arch"));
      echo ("<BR>");;
  ?>
  ```
  ```

PHP e Extreme Programming

PHP e XP

- XP não é o nome de um produto Microsoft, mas sim o nome de uma nova metodologia de engenharia de software.
- XP significa "eXtreme Programming" e busca desenvolver sistemas de alta qualidade em tempo reduzido pela aderência a um conjunto pequeno de boas práticas.
- O XP vem demonstrando resultados impressionantes e recebendo suporte tanto do mundo proprietário quando do mundo livre!

Princípios do XP

- Participação ativa do usuário como parte da equipe
- Iterações pequenas, no máximo 3 semanas, que sempre entregam algo funcional para o usuário
- Programação em pares (dois desenvolvedores para um micro) o tempo todo
- Integrações totais várias vezes ao dia
- Testar tudo, o tempo todo, com apoio da automação
- Os planos de teste precedem a programação das classes / módulos

O Xunit

- A metodologia XP depende fortemente da existência de mecanismos para testes automatizados, pois:
  - Todos os testes de unidade devem completar 100% para que uma integração seja bem sucedida.
  - Todos os testes de funcionalidade devem completar 100% para que uma interação seja considerada completa.
- O Xunit é um framework para o desenvolvimento, execução e tabulação dos testes

O PhpUnit

- O Xunit foi portado para diversas linguagens de programação, como o PHP.
- Um conjunto de testes é uma subclasse de TestCase, que executa código das classes do programa e verifica se os resultados são os esperados.
- Instâncias de TestCase são adicionadas a TestSuite, que executa os testes e tabula os resultados.
- TestSuite podem conter outros TestSuite de modo a facilitar a organização dos testes em hierarquias

Exemplo de TestCase

```
class TestesContaCorrente extends TestCase
{
    function TestesContaCorrente ($nome) {
        $this->TestCase ($nome);
    }
    function testaDeposito () {
        $conta = newContaCorrente (1000.0);
        $conta->deposito (100.0);
        $this->assertEquals ($conta->saldo, 1100.0);
    }
    function testaSaque () {
        $conta = newContaCorrente (1000.0);
        $conta->saque (2000.0);
        $this->assertEquals ($conta->saldo, 1000.0);
    }
}
```

Sobre um TestCase

- Bons TestCases devem incluir não apenas testes sobre o que deve ser o funcionamento correto de uma classe, mas também testes sobre situações de erro que a classe deve ser capaz de tratar ou detectar para que a aplicação seja robusta!

Exemplo de TestSuite

- <?php

```
require "phpunit.php";

// ... definição dos TestCases, provavelmente include

$suite = new TestSuite;
$suite->addTest (new TestSuite ("TestesContaCorrente"));

// não é preciso fazer mais nada para que os testes sejam
// executados e tabulados em uma página HTML

?>
```

PEAR, GTK+ e Xunit

- O GTK+ não segue o modelo (de destrutores e excesões) definido pelo PEAR
- O PhpUnit está sendo incorporado (e adaptado) ao PEAR e já aparece nas versões do PHP posteriores à 4.2.0

FIM

Referências

- www.php.net gtk.php.net
- www.gtk.org wingtk.sourceforge.net
- www.mysql.com
- phpunit.sourceforge.net
- www.xprogramming.com
- www.DevShed.com
- www.phpbuilder.com
- www.sourceforge.net

Perguntas

- Dúvidas:
  fernando@lozano.eti.br
- Palestras, artigos e Apostilas:
  www.lozano.eti.br
- Livro:
  Java em GNU/Linux
  www.altabooks.com.br