WBA0183_v2.0

PROGRAMAÇÃO PHP

Thiago Salhab Alves

Programação PHP

1ª edição

Londrina
Editora e Distribuidora Educacional S.A.
2019

Dados Internacionais de Catalogação na Publicação (CIP)

Alves, Thiago Salhab
A474p Programação PHP/ Thiago Salhab Alves, – Londrina:
Editora e Distribuidora Educacional S.A. 2019.
110 p.
ISBN 978-85-522-1609-4
1. Animações e ferramentas virtuais. 2. Paradigma Orientado a Objetos. I. Alves, Thiago Salhab. Título.

CDD 004

Thamiris Mantovani CRB: 8/9491

2019
Editora e Distribuidora Educacional S.A.
Avenida Paris, 675 – Parque Residencial João Piza
CEP: 86041-100 — Londrina — PR
e-mail: editora.educacional@kroton.com.br
Homepage: http://www.kroton.com.br/

SUMÁRIO

Apresentação da disciplina 5

Introdução à programação PHP 6

Tipos no PHP 19

Constantes, variáveis e operadores 32

Estruturas de controle 49

Funções 63

PHP e orientação a objetos 76

PHP e MySQL 93

Apresentação da disciplina

A disciplina de Programação PHP apresenta os elementos essenciais para o desenvolvimento de páginas dinâmicas, sendo o PHP uma das principais linguagens utilizadas para a programação de páginas web. Vamos conhecer os conteúdos que serão explorados em nossas aulas!

No Tema 1 serão trabalhados os conceitos básicos da linguagem de programação PHP, procedimentos para instalação do software EasyPHP e uso de marcadores e comentários. No Tema 2 serão trabalhados os tipos de dados na linguagem de programação PHP, explorando os tipos escalares, compostos e especiais. No Tema 3 serão trabalhadas as contantes, variáveis e operadores aritméticos, relacionais e lógicos no PHP. No Tema 4 serão trabalhadas as estruturas de controle no PHP (if e if else if), as estruturas de repetição (while, do/while, for e foreach) e as estruturas break, continue, return e goto. No Tema 5 serão trabalhadas a criação e utilização de funções no PHP, o uso de argumentos e o retorno de valores de uma função e funções recursivas e anônimas. No Tema 6 serão trabalhados os conceitos de orientação a objetos (classes, objetos, atributos e métodos), o uso da orientação a objetos no PHP e o controle de exceções. Para finalizar, no Tema 7 serão trabalhados a instalação do MySQL, a criação de um banco de dados usando SQL e a criação de classes de conexão do PHP com o MySQL.

Introdução à programação PHP

Autor: Thiago Salhab Alves

Objetivos

- Aprender os conceitos básicos do PHP;
- Aprender a instalar o software EasyPHP e testar o funcionamento do PHP;
- Aprender a utilizar marcadores de comandos e comentários.

1. Introdução à programação PHP

Prezado aluno, você já parou para pensar como seriam as páginas Web sem uma atualização dinâmica? Como seria o dia a dia de um editor de conteúdos se tivesse que realizar atualizações e substituir o conteúdo das páginas tendo que tirá-las do ar? Certamente, o processo de criação de páginas dinâmicas não seria possível. Para que isso seja possível, utiliza-se o PHP.

Esta leitura irá apresentar os conceitos básicos do PHP. Você aprenderá sobre o processo de instalação do EasyPHP e testar o funcionamento do PHP. Você aprenderá também a utilizar os marcadores de comando e comentários. Uma boa aula e bom trabalho!

PARA SABER MAIS

PHP é uma linguagem de script, *open source* (código aberto), de uso geral, muito utilizada para o desenvolvimento de aplicações Web integradas com códigos HTML.

ASSIMILE

Diferentemente de outras linguagens, tais como: C, C++ e Perl, nas quais é necessário escrever vários comandos para produzir uma simples página HTML, em PHP você precisa apenas juntar o HTML aos comandos PHP, demarcando-os por meio de tags (marcadores) especiais.

2. Programação PHP

O PHP, segundo Soares (2013), se refere a *Hypertext Preprocessor* (pré-processador de hipertexto), sendo uma poderosa linguagem de

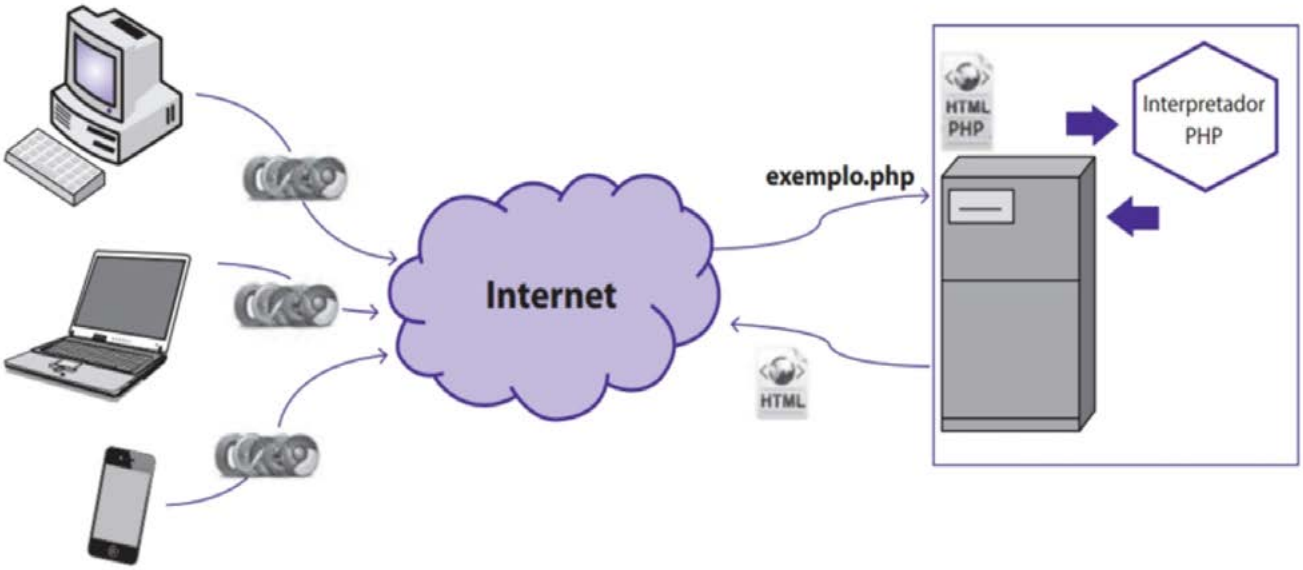
programação *open source* (código fonte de utilização livre), conhecida mundialmente e utilizada no ambiente Web.

De acordo com Miletto e Bertagnolli (2014), o PHP é uma linguagem criada em 1995, por Ramus Ledorf, sendo usada na criação de scripts que são interpretados em um servidor Web, tendo como pré-requisito para seu funcionamento o servidor ter o interpretador PHP devidamente configurado. Os scripts também podem ser executados localmente via linha de comando, através de um interpretador.

A potência da linguagem para proporcionar dinamismo às páginas Web, segundo Miletto e Bertagnolli (2014), fez com que seu uso atingisse um considerável crescimento nos últimos anos, sendo utilizada em aproximadamente 39% dos sites disponíveis na internet. O PHP se destaca por sua apresentação em conjunto com as marcações da linguagem HTML (*Hipertext Markup Language*), possibilitando a adição de dinamicidade às páginas desenvolvidas nessa linguagem.

Para sua identificação pelo servidor Web, os trechos que serão interpretados como scripts em PHP precisam usar delimitadores (TAGs) iniciais. As tags <?php e ?> são utilizadas, respectivamente, como início e término de um bloco em PHP, diferenciando-as, por exemplo, do HTML. A Figura 1 apresenta a forma como as linguagens trabalham em conjunto utilizando a arquitetura cliente-servidor. O cliente, através de seu navegador Web, solicita acesso ao script (página Web) 'exemplo.php'. O servidor irá procurar pelo arquivo e passará a interpretar o código apresentado, processando os trechos dentro das TAGs de identificação do PHP. Após esse processamento, o servidor enviará ao cliente (navegador do usuário) o resultado em conjunto com os demais códigos em HTML. Assim, do lado cliente, não há conhecimento referente aos códigos previamente existentes na linguagem PHP, pois as páginas são recebidas devidamente processadas pelo servidor.

Figura 1 – Interpretação do PHP



Fonte: MILETTO e BERTAGNOLLI (2014, p. 163).

O PHP, segundo Miletto e Bertagnolli (2014), se destaca por oferecer também suporte a um grande número de bancos de dados, como Firebird/ Interbase, MySQL, PostgreSQL, IBM DB2, dentre vários outros. Considere o exemplo da Figura 2, na qual é apresentada a inserção de um código PHP juntamente a um código HTML. O HTML é uma linguagem de marcação de hipertexto. O início e término de um código HTML é realizado pela tag <html> para o início e </html> para o término. O cabeçalho da página Web pode receber um título para sua descrição. As tags utilizadas são <head> <title> e </title> </ head>. No exemplo da Figura 2, o título utilizado foi PHP 5 – Guia do Programador. O conjunto de tags <body> </body> indicam o início e término do corpo de código que será utilizado na página Web. Todos os textos, figuras e demais elementos gráficos são utilizados através dessa tag. O código PHP foi incluído através da tag <?php e ?>, sendo usado, respectivamente, para início e término do código PHP. Neste exemplo, o comando echo é utilizado para imprimir uma String que contém o texto “Exemplo de um programa PHP”.

Figura 2 – Exemplo de utilização do PHP

```
<html>
   <head>
      <title>PHP 5 - Guia do Programador</title>
   </head>
   <body>
        <?php
           echo "Exemplo de um programa PHP";
        ?>
   </body>
</html>
```

Fonte: SOARES (2013, p. 28).

A criação de código PHP pode ser realizada em qualquer editor de texto, porém alguns editores estão disponíveis para facilitar escrever os scripts, apresentando algumas características que facilitam o trabalho do desenvolvedor. Segundo Miletto e Bertagnolli (2014), algumas dessas características são:

- Função de autocomplemento: é um recurso que auxilia o programador a recordar códigos (funções e procedimentos), autocompletando o código com base em trechos iniciais de sua sintaxe. Por exemplo, ao digitar inicialmente a tag <?, o editor apresenta a opção de autocomplemento <?php e faz o fechamento com a tag ?>.

- Verificador de sintaxe: verifica após a digitação da tag se ela está correta ou precisa ser corrigida. A verificação ocorre enquanto o programador está digitando seu código.

- Possibilidade de customizar templates: alguns editores apresentam a opção de utilizar templates (modelo de páginas) e realizar sua customização (alterações).

- Depurador (*debugger*): é um programa que analisa o código de outro programa para testar e encontrar erros no código, permitindo aos desenvolvedores melhorar seus programas e entender melhor seu funcionamento.

- Sistema para transferência de arquivos para o servidor: permite enviar os códigos produzidos para o servidor.

Para o desenvolvimento de código PHP, há alguns editores de texto que atendem as características mencionadas anteriormente e que são gratuitos:

- Netbeans IDE.
- Eclipse PDT.
- Bluefish.
- Notepad++.
- Sublime Text.
- PHP Editor.

2.1 EasyPHP

O software EasyPHP é um software da categoria WAMP, que é um pacote de soluções para Windows, Apache, MySQL e PHP, instalando um servidor WEB Apache, um módulo para programação em PHP e o MySQL (SGBD). A grande vantagem de se utilizar o EasyPHP na plataforma Windows é poder trabalhar com programação PHP sem se preocupar com os passos necessários para se configurar um servidor Apache e servidores de Banco de Dados.

O EasyPHP é instalado através do Devserver, que apresenta um ambiente de desenvolvimento completo e pronto para uso. É portátil, modular, totalmente configurável e fácil de atualizar e estender. Permite que se configure um servidor local com as mesmas características do servidor de produção, podendo realizar o desenvolvimento em qualquer lugar. Alguns recursos:

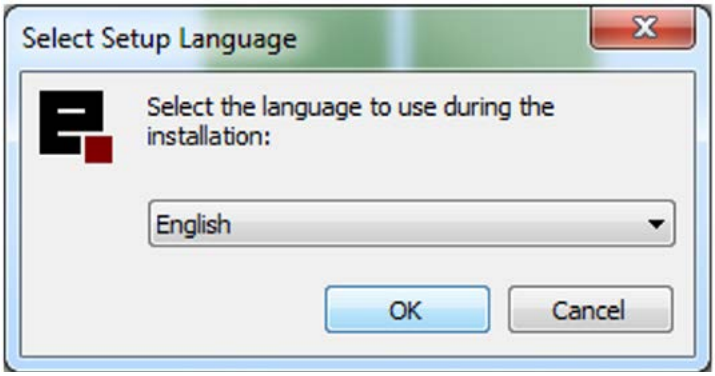
- Pronto para usar: baixar, instalar e codificar.
- Portátil: usb, cartões de memória e HD’s externos.
- Totalmente configurável: portas, fuso horário e extensões.
- Pode ser utilizado por iniciantes e especialistas.
- Permite iniciar, parar e reiniciar servidores.
- Acesso direto a arquivos e pastas.
- Arquivos de configuração, log de erros e log de acesso.
- Modular: adicionar aplicativos e versões.

Para a instalação do EasyPHP, siga os seguintes passos:

1. Acesse o site do EasyPHP[1].
2. Selecione a versão adequada de seu sistema operacional Windows. Para Windows 7/8/10 realizar o download do Devserver 17.0 e para Windows XP a versão Devserver 17.0 lite (essa versão não suporta o PHP 5.5.x, 5.6.x, 7.x).
3. Baixe o EasyPHP-Devserver-17.0-setup e inicie a instalação com dois cliques no ícone, conforme a Figura 3. Selecione o idioma inglês (english) e clique no botão OK.

[1] https://www.easyphp.org

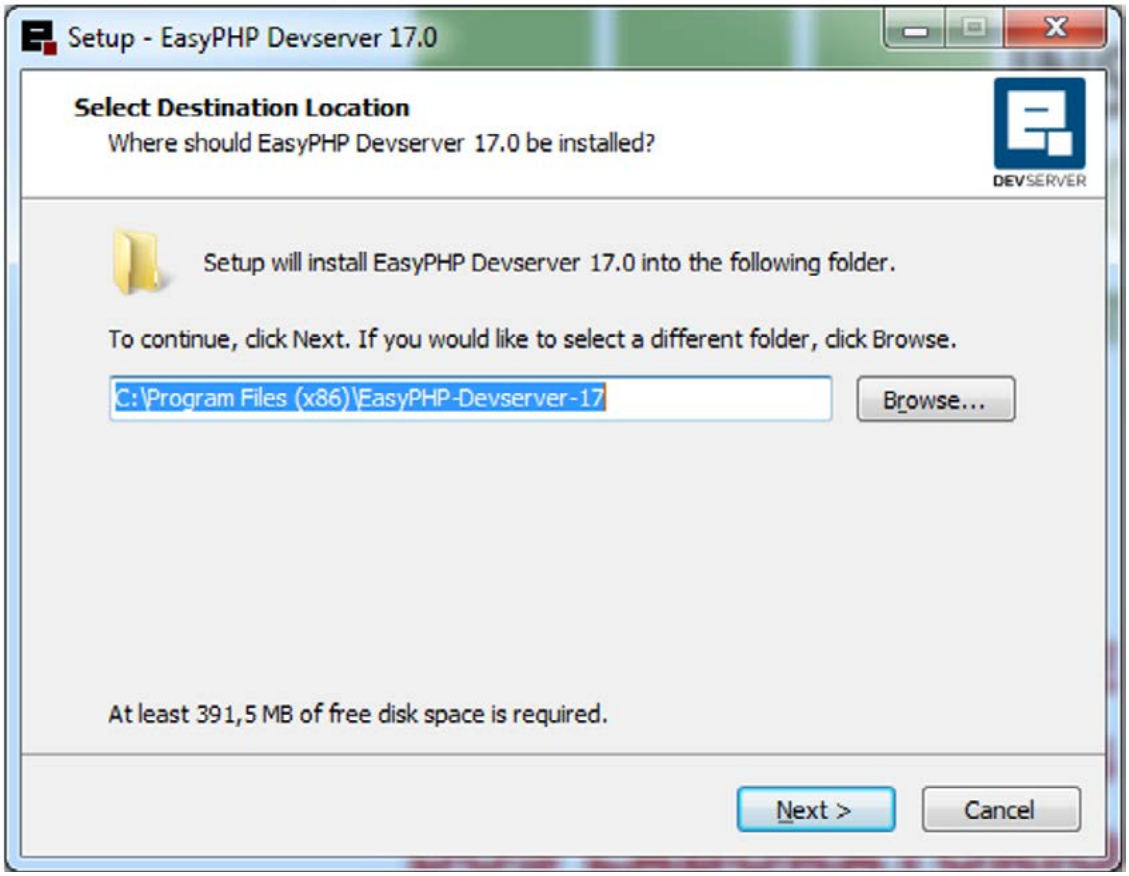
Figura 3 – Tela inicial de instalação do EasyPHP



Fonte: o autor.

4. Determine o local em que o EasyPHP Devserver 17.0 será instalado. Por padrão, o local de instalação é C:\Program Files (x86)\EasyPHP-Devserver-17 e clique no botão Next, conforme a Figura 4.

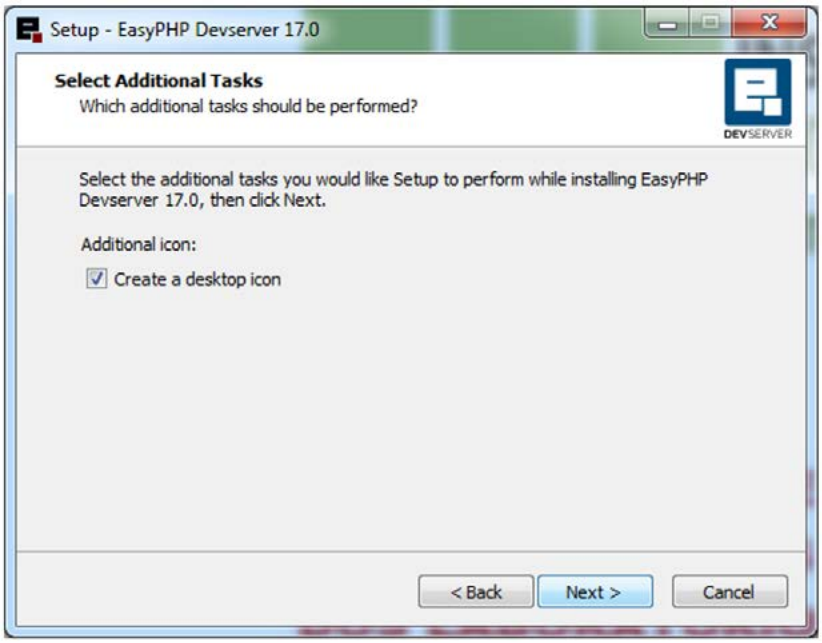
Figura 4 – Tela de seleção do local de instalação do EasyPHP



Fonte: o autor.

5. Selecione a caixa para criar um ícone na Área de Trabalho e clique em Next, conforme a Figura 5.

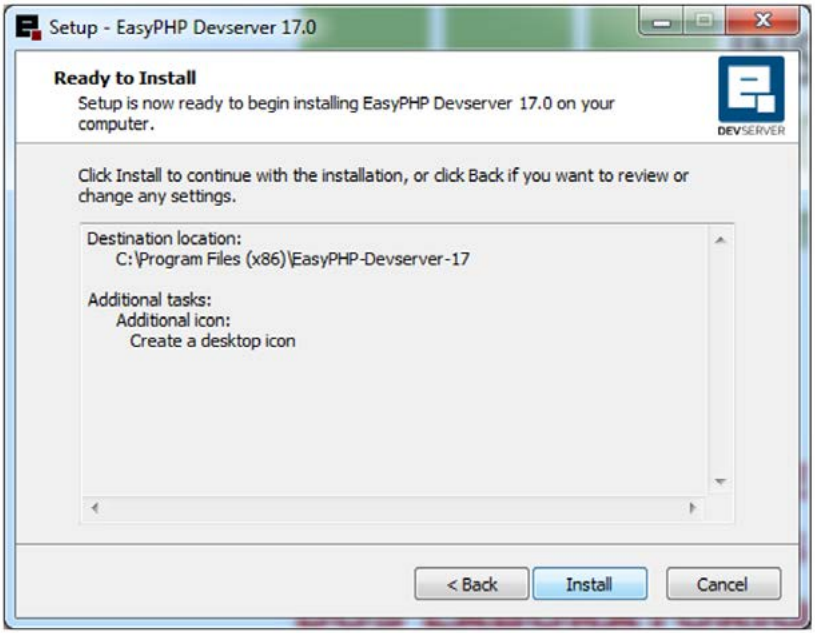
Figura 5 – Tela para criar ícone inicial para acesso ao EasyPHP



Fonte: o autor.

6. Proceda com a instalação clicando em Install, conforme a Figura 6.

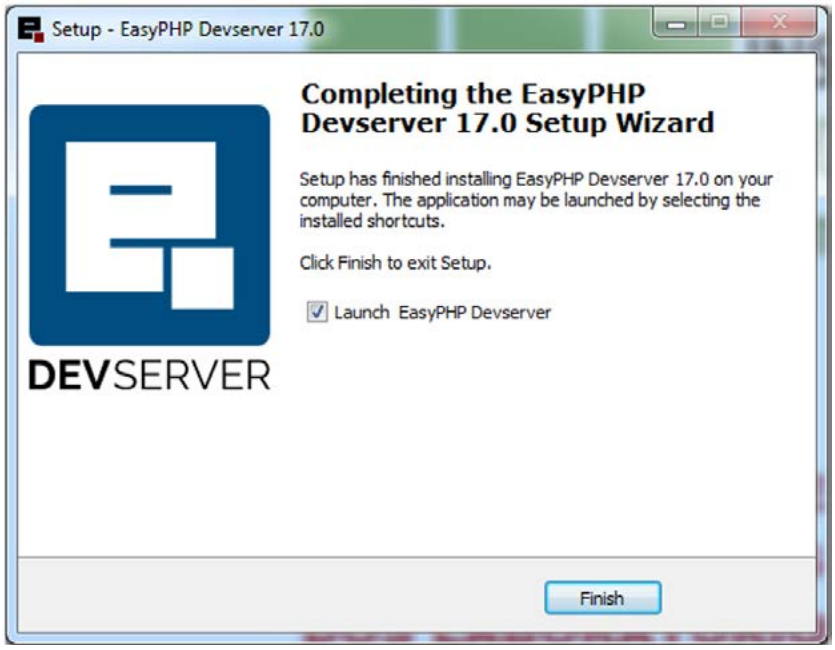
Figura 6 – Tela de instalação do EasyPHP



Fonte: o autor.

7. Finalize a instalação clicando no botão Finish, conforme a Figura 7.

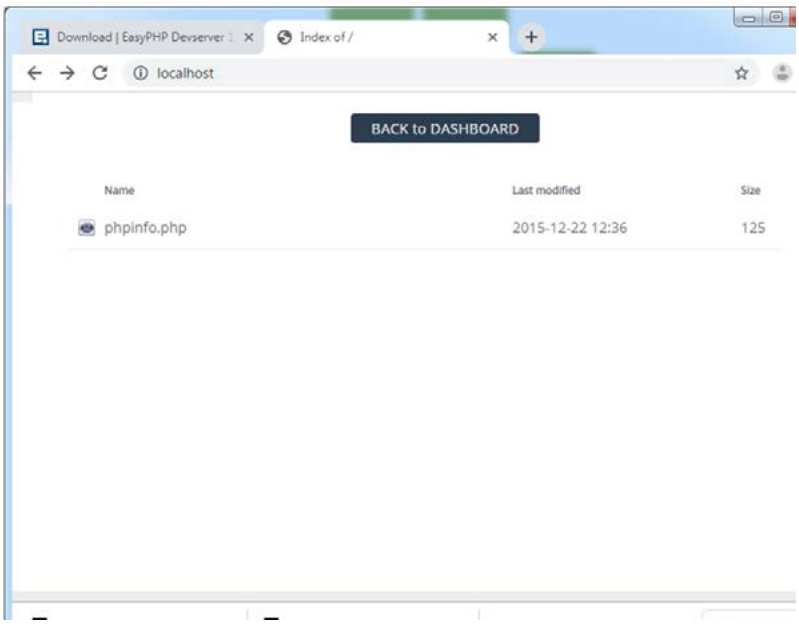
Figura 7 – Tela de finalização da instalação do EasyPHP



Fonte: o autor.

8. Para testar o funcionamento do EasyPHP, abra o seu navegador e digite localhost, conforme a Figura 8.

Figura 8 – Teste de funcionamento do EasyPHP



Fonte: o autor.

Dessa forma, você aprendeu os conceitos básicos do PHP e o processo de instalação do EasyPHP, bem como testar seu funcionamento. Você aprendeu também a utilizar os marcadores de comando e comentários.

TEORIA EM PRÁTICA

Uma empresária nacional, do ramo de moda, gostaria de criar um blog para poder divulgar seus produtos.
Dado a grande concorrência do setor de moda, a empresária está tendo dificuldades para divulgar seus lançamentos aos clientes e gostaria que pudesse ser criado um blog dinâmico, no qual pudesse realizar as postagens de fotos de seus produtos, incluir vídeos, permitir que os clientes comentem suas postagens, conseguindo assim atingir novos públicos e maximizar suas vendas.
Hoje, a empresária conta com uma loja física e envia as promoções para seus clientes via aplicativos de mensagens instantâneas e e-mails marketing. Como podemos auxiliar a empresária na escolha de uma linguagem de programação para que seja possível a construção desse blog de modas?

VERIFICAÇÃO DE LEITURA

1. O PHP é uma linguagem criada em 1995, por Ramus Ledorf, sendo usada na criação de scripts que são interpretados em um servidor Web. Assinale a alternativa que apresenta, corretamente, as siglas de PHP:

   a. *PHP – Hypertext Preprocessor.*

   b. *PHP – Home Processor.*

   c. *PHP – Product Home Processor.*

d. *PHP – Hypertext Processor.*

e. *PHP – Personal Preprocessor.*

2. O PHP é uma poderosa linguagem de programação *open source* (código fonte de utilização livre), conhecida mundialmente e utilizada no ambiente Web. Para que seus comandos possam ser interpretados, são utilizados delimitadores. Assinale a alternativa que apresenta, corretamente, o nome desses delimitadores.

   a. Scripts.

   b. Códigos.

   c. Tags.

   d. Sentenças.

   e. Blocos.

3. O software EasyPHP é um software que apresenta soluções para Windows, Apache, MySQL e PHP, instalando um servidor WEB Apache, um módulo para programação em PHP e o MySQL (Sistema Gerenciador de Banco de Dados). Assinale a alternativa que apresente, corretamente, a que categoria o EasyPHP pertence:

   a. LAMP.

   b. XAMP.

   c. WAMP.

   d. WAMT.

   e. XAMT.

Referências bibliográficas

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

SOARES, Walace. **PHP 5:** conceitos, programação e integração com banco de dados. 7. ed. São Paulo: Érica, 2013.

Gabarito

Questão 1 – Resposta: A

Resolução: As siglas do PHP se referem a Hypertext Preprocessor (Pré-processador de hipertexto), linguagem usada na criação de scripts que são interpretados em um servidor Web

Feedback de reforço: refere-se a um pré-processador de hipertexto.

Questão 2 – Resposta: C

Resolução: O PHP utiliza para o funcionamento de seus comandos alguns delimitadores, que são chamados de tags, fazendo com que os comandos possam ser interpretados .

Feedback de reforço: Recebe o mesmo nome dos delimitadores usados em HTML.

Questão 3 – Resposta: C

Resolução: O software EasyPHP é um software de categoria WAMP que apresenta soluções para Windows, Apache, MySQL e PHP, instalando um servidor WEB Apache, um módulo para programação em PHP e o MySQL (Sistema Gerenciador de Banco de Dados).

Feedback de reforço: Siglas para Windows, Apache, MySQL e PHP.

Tipos no PHP

Autor: Thiago Salhab Alves

Objetivos

- Aprender os tipos escalares do PHP;
- Aprender os tipos compostos do PHP;
- Aprender os tipos especiais do PHP.

1. Tipos no PHP

Prezado aluno, você já parou para pensar como seria a utilização de variáveis nos programas se não houvessem tipos de dados? Como seria o dia a dia de um editor de conteúdos se fizesse a definição de suas variáveis em tipos errados? Certamente, o processo do trabalho seria dificultado. Para isso, o PHP trabalha com diversos tipos de dados e faz a atribuição automática do tipo de acordo com os valores utilizados pelas variáveis.

Esta leitura irá apresentar os tipos de dados utilizados no PHP. Você aprenderá sobre os tipos de dados escalares, compostos e especiais. Uma boa aula e bom trabalho!

PARA SABER MAIS

No PHP é possível converter uma variável de um tipo para outro. Para isso são utilizados conversores de tipos: (int), (integer) – converte no tipo inteiro; (bool), (boolean) – converte em booleano; (float), (double), (real) – converte em ponto flutuante; (string) – converte em string; (array) – converte em array e (object) – converte em objeto.

ASSIMILE

É possível, segundo Soares (2013), saber o tipo de uma variável através da função gettype(). O resultado é a descrição do tipo da variável. A sintaxe para sua utilização é string gettype(variável).

1.1 Tipos de dados no PHP

Segundo Soares (2013), o PHP possui oito tipos básicos de dados que estão divididos em três grupos:

- Escalares: Inteiros (int), Ponto Flutuante (float, double ou real), String e Booleanos;
- Compostos: Arrays e Objetos;
- Especiais: Recursos e Nulo (Null).

Um dos recursos utilizados pelo PHP, segundo Soares (2013), é a não preocupação do desenvolvedor em definir o tipo da variável, pois o PHP faz isso automaticamente. Uma variável é uma área reservada da memória responsável por receber valores que serão utilizados. A definição de uma variável ocorre incluindo o símbolo de cifrão (**$**) antes do nome da variável. A Figura 1 apresenta algumas variáveis que possuem seus tipos definidos automaticamente, através da atribuição de valores iniciais. A variável i recebe como atribuição o valor **10** e, dessa forma, o PHP a definirá como tipo inteiro. A variável **nome** recebe como atribuição o nome **Walace** e, portanto, o PHP a definirá como tipo String. A variável **falso** recebe como atribuição o valor **FALSE** e, portanto, será definida como Booleano. Por fim, a variável **valor** recebe como atribuição o valor 100.50 e, portanto, será definida automaticamente para Ponto Flutuante.

Figura 1 – Tipos do PHP

```
<?php
 $i = 10; // Inteiro
 $nome = "Walace"; // String
 $falso = FALSE; // Booleano
 $valor = 100.50; /// Ponto flutuante
?>
```

Fonte: SOARES (2013, p. 46).

De acordo com Saraiva e Barreto (2018), o PHP é uma linguagem de programação que apresenta tipagem fraca, o que significa que é o PHP que se encarrega de alterar os tipos de dados da variável quando for necessário, conforme a operação que estiver sendo realizada. Isso faz com que a tipagem se torne dinâmica, pois não há necessidade de o programador informar, de forma fixa, se o tipo de cada variável é string, inteiro, ponto flutuante, objeto, etc.

Segundo Miletto e Bertagnolli (2014), o PHP não requer que as variáveis sejam declaradas antes de sua utilização, fato que difere de outras linguagens tradicionais. As variáveis são criadas na primeira vez que um valor for atribuído a ela. A variável também não precisa ser inicializada no PHP, mas essa prática pode ser recomendável em situações em que se deseja ter a garantia de que a variável não foi utilizada anteriormente.

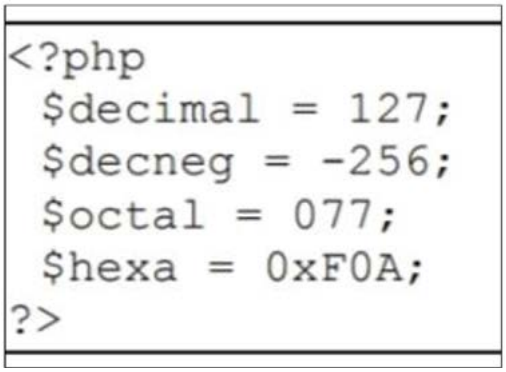
1.2 Inteiro

O tipo Inteiro, segundo Soares (2013), é utilizado por qualquer número sem decimais, positivo ou negativo, podendo ser representado na base decimal, hexadecimal ou octal. Para se definir um inteiro na base octal, o número deve ser precedido de 0 (zero), e para definir um número na base hexadecimal, o número deve ser precedido de 0x. O tamanho máximo do tipo inteiro vai depender da plataforma em que o PHP está sendo executado, mas em geral equivale a 32 bits. No caso de especificar um número inteiro além dos limites suportados pelo PHP, o mesmo será definido como ponto flutuante, evitando assim o overflow (estouro de buffer).

A Figura 2 ilustra o uso das variáveis do tipo Inteiro. A variável **decimal** recebe como atribuição o valor **127**, sendo definida automaticamente para Inteiro na base decimal. A variável **decneg** recebe como atribuição o valor **-256**, sendo definida automaticamente como Inteiro negativo. A variável **octal** recebe como atribuição o valor **077**, sendo definida automaticamente como Inteiro octal, pois o valor é precedido de 0 (zero).

Por fim, a variável hexa recebe o valor 0xF0A, sendo definida automaticamente como hexadecimal.

Figura 2 – Exemplo de utilização de Inteiro no PHP



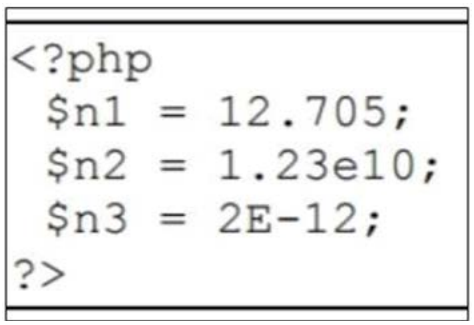
```
<?php
 $decimal = 127;
 $decneg = -256;
 $octal = 077;
 $hexa = 0xF0A;
?>
```

Fonte: SOARES (2013, p. 47).

1.3 Ponto flutuante

Segundo Soares (2013), o tipo ponto flutuante apresenta precisão dupla ou um número real. O tamanho máximo do ponto flutuante depende da plataforma em que o PHP está sendo executado, mas em geral é algo em torno de 1.8e308 com uma precisão de 14 dígitos decimais, equivalente a 64 bits do padrão IEEE. Números inteiros muito grandes também serão considerados como ponto flutuante. A Figura 3 ilustra o exemplo de utilização de variáveis do tipo ponto flutuante. As variáveis **n1**, **n2** e **n3** são inicializadas com os valores 12.705, 1.23e10 e 2E-12, respectivamente, que se referem ao tipo ponto flutuante.

Figura 3 – Exemplo de utilização de ponto flutuante no PHP



```
<?php
 $n1 = 12.705;
 $n2 = 1.23e10;
 $n3 = 2E-12;
?>
```

Fonte: SOARES (2013, p. 49).

1.4 String

De acordo com Soares (2013), uma String é uma cadeia de caracteres e, para o PHP, um caractere é exatamente um byte. Uma String pode ser definida em três formatos diferentes:

- Aspas Simples: para definir uma string com aspas simples, basta para isso delimitar qualquer texto com aspas simples. A Figura 4 ilustra o exemplo da definição da variável **s1** recebendo a String **PHP 5 – Guia do Programador**. O uso de aspas simples para Strings impede que conteúdos de variáveis sejam utilizados juntamente da String (todo o conteúdo será considerado como String).

Figura 4 – Exemplo de utilização de variável String com Aspas Simples

```
<?php
 $s1 = 'PHP 5 - Guia do Programador';
?>
```

Fonte: SOARES (2013, p. 49).

- Aspas Duplas: o PHP aceita que uma string seja delimitada pelo caractere aspas duplas. Ao contrário das aspas simples, o uso das aspas duplas permite a impressão de conteúdo de variáveis na String (SOARES, 2013). A Figura 5 ilustra a definição de uma variável **s1** recebendo a String **PHP 5 – Guia do Programador** através das aspas duplas.

Figura 5 – Exemplo de utilização de variável String com Aspas Duplas

```
<?php
 $s1 = "PHP 5 - Guia do Programador";
?>
```

Fonte: SOARES (2013, p. 51).

Sintaxe Heredoc: uma outra maneira de utilizar strings no PHP, de acordo com Soares (2013), é através da sintaxe Heredoc, que faz uso de um caractere predefinido para delimitar uma string. Deve-se primeiramente utilizar três sinais de menor <<< e em seguida definir o delimitador (que deve estar presente no final da string) e informar o texto da string. O delimitador deve ser formado apenas por caracteres alfanuméricos e _ (*underscore*) e não deve ser iniciado por um caractere numérico. Assim como o uso de aspas duplas, a sintaxe Heredoc permite a impressão de conteúdo de variáveis na String. A Figura 6 ilustra o exemplo de uso de string através da sintaxe Heredoc. É definido o delimitador **EOL** precedido dos três sinais de menor <<< atribuindo para a variável **s1** os caracteres Este é um Exemplo da utilização do Formato heredoc no PHP para definir uma string.

Figura 6 – Exemplo de uso de variável String pela sintaxe Heredoc

```
<?php
   $s1 = <<< EOL
         Este é um Exemplo da utilização do
      Formato heredoc no PHP para definir
      Uma string
   EOL;
?>
```

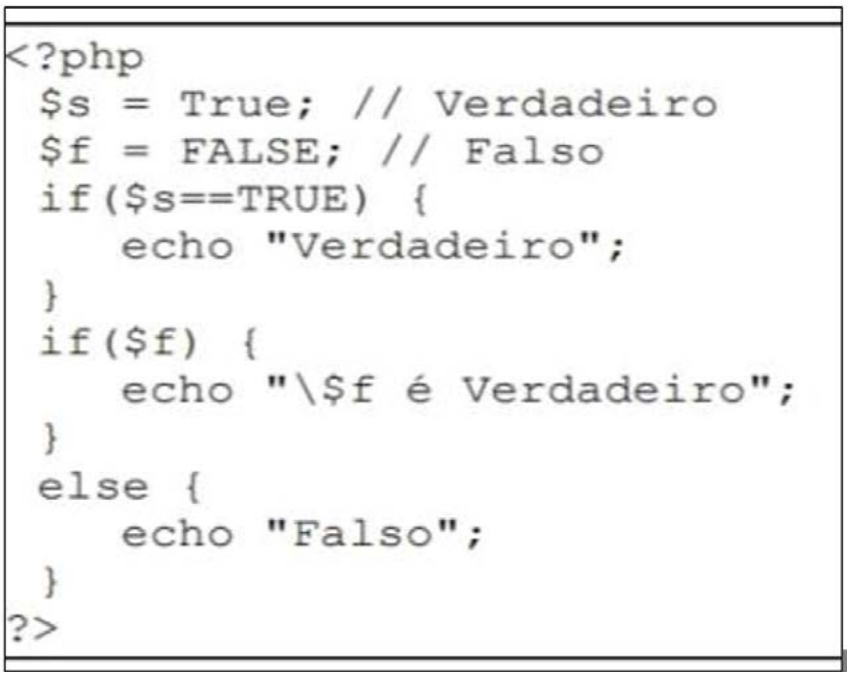
Fonte: SOARES (2013, p. 51).

1.5 Booleano

O tipo booleano é utilizado para se atribuir valores Verdadeiro (True) ou Falso (False). O tipo booleano não faz distinção entre letras maiúsculas de minúsculas, isto é, TRUE, True, true são iguais para o PHP. A Figura 7 ilustra um exemplo de declaração da variável **s** que é inicializada com True e **f** que é inicializada com FALSE. Dessa forma as variáveis são definidas como Booleanos. O if irá verificar o valor das variáveis. Se o conteúdo da variável s for igual a TRUE (lembrando que não há distinção entre letras maiúsculas e minúsculas), será impresso

Verdadeiro. O próximo if verifica o valor da variável f. Como f possui valor false, o bloco else (senão) é executado e será impresso Falso.

Figura 7 – Exemplo de uso de variável Booleano



```
<?php
 $s = True; // Verdadeiro
 $f = FALSE; // Falso
 if($s==TRUE) {
    echo "Verdadeiro";
 }
 if($f) {
    echo "\$f é Verdadeiro";
 }
 else {
    echo "Falso";
 }
?>
```

Fonte: SOARES (2013, p. 52).

1.6 Array

No PHP, de acordo com Soares (2013), arrays são mapas ordenados de chaves e valores, podendo atribuir a um elemento do array, uma chave e um valor. Através do Array é possível representar listas, pilhas, filas, etc. Um array é definido de forma explícita, na qual se utiliza o construtor array(), que possui a seguinte sintaxe:

- Array ([chave=>] valor, ...)

Se uma chave não for especificada, o PHP vai atribuir um índice numérico crescente ao elemento do array. A chave pode ser um número, representando um índice do array ou uma string (chamado array associativo, associa a chave ao valor informado). A Figura 8 ilustra o exemplo do uso de arrays. Há o array $a com os elementos 0 = “cliente 1”, 3 = “cliente 2” e 4 = “cliente 3” (o índice do segundo elemento do array

é forçado para 3, então os índices são 0, 3 e 4). O array $a2 é associativo, no qual as chaves são as strings “cliente 1”, “cliente 2” e “cliente 3”. O último array $a2 é multidimensional associativo.

Figura 8 – Exemplo de uso de Arrays

```
<?php
 $a = array("cliente 1", 3 => "cliente 2", "cliente 3");
 $a1 = array("cliente 1" => 1000, "cliente 2" => 5000, "cliente 3" => 0);
 $a2 = array("cliente 1" =>
                array("endereço" => "Rua x, no. 10",
                      "bairro" => "Vila Maria",
                      "cidade" => "São Paulo"),
             "cliente 2" =>
                array("endereço" => "Rua y, no. 102 ",
                      "bairro" => "Glória",
                      "cidade" => "Vila Velha"),
             "cliente 3" =>
                array("endereço" => "Rua z, no. 2 ",
                      "bairro" => "Centro",
                      "cidade" => "Curitiba"));
?>
```

Fonte: SOARES (2013), p. 53.

1.7 Objetos

No PHP, de acordo com Soares (2013), um objeto é uma instância de uma classe, isto é, a vinculação de uma classe, com um conjunto de propriedades e métodos. O operador new é utilizado para instanciar objetos. A Figura 9 apresenta uma classe chamada novaClasse, que contém uma função imprime, que exibe uma mensagem. A variável n realiza a instância de um novo objeto de novaClasse através do operador new. Após a instanciação, o objeto pode realizar a chamada da função imprime.

Figura 9 – Exemplo de uso de objetos

```
<?php
class novaClasse {
    function imprime( ) {
        echo "Esta é um novo objeto de novaClasse";
    }
}
$n = new novaClasse;
$n -> imprime( );
?>
```

Fonte: o autor

1.8 Recursos

Os recursos, segundo Soares (2013), são variáveis especiais no PHP, que referenciam recursos externos ao PHP. Os recursos são criados e utilizados por funções especiais, por exemplo, manipulação de bancos de dados, imagens, arquivos, etc.

1.9 Nulo

O tipo nulo, de acordo com Soares (2013), representa uma variável sem nenhum valor. O tipo nulo aceita apenas o valor NULL:

```
<?php
    $nulo = NULL;
?>
```

O PHP considera uma variável como nula em uma das seguintes situações:

- Se a variável receber o valor NULL.
- Se a variável não tiver sido criada ainda.

Dessa forma, você aprendeu sobre os tipos de dados utilizados no PHP, sendo os tipos de dados escalares, compostos e especiais.

TEORIA EM PRÁTICA

Uma empresária nacional, do ramo de moda, gostaria de criar um blog para poder divulgar seus produtos. Dado a grande concorrência do setor de moda, a empresária está tendo dificuldades para divulgar seus lançamentos aos clientes e gostaria que pudesse ser criado um blog dinâmico, onde pudesse realizar as postagens de fotos de seus produtos, incluir vídeos, permitir que os clientes comentem suas postagens, conseguindo assim atingir novos públicos e maximizar suas vendas. Hoje, a empresária conta com uma loja física e envia as promoções para seus clientes via aplicativos de mensagens instantâneas e e-mails marketing. Como podemos auxiliar a empresária na escolha das variáveis que serão utilizadas para a divulgação das informações de seus produtos?

VERIFICAÇÃO DE LEITURA

1. É utilizado por qualquer número sem decimais, positivo ou negativo, podendo ser representado na base decimal, hexadecimal ou octal. Assinale a alternativa que apresenta, corretamente, o tipo de dado da afirmação:

   a. Inteiro

   b. Ponto Flutuante

   c. String

d. Booleano

e. Array

2. Este tipo de dado se caracteriza por apresentar precisão dupla e o tamanho máximo depende da plataforma em que o PHP está sendo executado, mas em geral é algo em torno de 1.8e308 com uma precisão de 14 dígitos decimais, equivalente a 64 bits do padrão IEEE. Assinale a alternativa que apresenta, corretamente, o tipo de dado descrito:

   a. Inteiro

   b. Ponto Flutuante

   c. String

   d. Booleano

   e. Array

3. Este tipo de dado é utilizado para atribuir valores Verdadeiro (True) ou Falso (False), não fazendo distinção entre letras maiúsculas e minúsculas. Assinale a alternativa que apresente, corretamente, o tipo de dado descrito:

   a. Inteiro

   b. Ponto Flutuante

   c. String

   d. Booleano

   e. Array

Referências bibliográficas

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

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.

Gabarito

Questão 1 – Resposta: A

Resolução: O tipo Inteiro é utilizado por qualquer número sem decimais, positivo ou negativo, podendo ser representado na base decimal, hexadecimal ou octal.

Feedback de reforço: Se refere aos números sem decimais.

Questão 2 – Resposta: B

Resolução: O ponto flutuante precisão dupla e o tamanho máximo depende da plataforma em que o PHP está sendo executado, mas em geral é algo em torno de 1.8e308 com uma precisão de 14 dígitos decimais, equivalente a 64 bits do padrão IEEE.

Feedback de reforço: Se refere aos números reais.

Questão 3 – Resposta: D

Resolução: Booleano é utilizado para se atribuir valores Verdadeiro (True) ou Falso (False), não fazendo distinção entre letras maiúsculas e minúsculas.

Feedback de reforço: Só aceita valores True ou False.

Constantes, variáveis e operadores

Autor: Thiago Salhab Alves

Objetivos

- Aprender sobre constantes no PHP;
- Aprender sobre variáveis no PHP;
- Aprender sobre operadores no PHP.

1. Introdução

Prezado aluno, você já parou para pensar como seria o desenvolvimento de programas em PHP se não houvessem constantes, variáveis e operadores? Certamente, o processo de trabalho seria dificultado. Para isso, o PHP trabalha com constantes, variáveis e operadores utilizados juntamente com as variáveis.

Esta leitura irá apresentar constantes utilizados no PHP. Você aprenderá sobre variáveis e operadores usados no PHP. Uma boa aula e bom trabalho!

PARA SABER MAIS

O PHP disponibiliza, segundo Soares (2013), uma forma de acessar as variáveis enviadas pelo browser para o servidor diretamente, ou seja, se temos no formulário web um campo com o nome “CODIGO” (<input type= “text” name= “CODIGO” value= “10”>). O script PHP que receber o formulário pode acessar essa variável diretamente ($CODIGO). Esse método não é recomendado e é muito inseguro, permitindo que alguém envie textos maliciosos. Usar $_POST[“CODIGO”].

ASSIMILE

É possível, segundo Soares (2013), saber o tipo de uma variável através da função gettype(). O resultado é a descrição do tipo da variável. A sintaxe para sua utilização é string gettype(variável).

2. Constantes e variáveis no PHP

Segundo Soares (2013), uma constante é o identificador de um valor no PHP. Quando uma constante é especificada, seu valor não pode ser alterado ou removido. Para se definir uma constante é utilizado o comando define(), que possui a seguinte sintaxe:

- define (<nome>, <valor>, [<case insensitive>]);

O nome indica o nome da constante, que deve começar com uma letra ou caractere subscrito (_) seguido de letras, números ou caracteres subscritos. Utiliza-se a convenção que o nome de uma constante possua apenas letras maiúsculas. As constantes podem possuir como valor somente tipos escalares, sendo aceitos os tipos inteiro, string, pontos flutuantes e booleanos.

O PHP, segundo Soares (2013), considera que as constantes são case sensitive, diferenciando letras maiúsculas de minúsculas. Para que as constantes possam ser case insensitive (não diferenciando letras minúsculas de maiúsculas), o terceiro parâmetro do comando define() deve conter o valor TRUE. A Figura 1 ilustra a definição de duas constantes. A primeira constante, “DADOS”, foi definida com valor 10 e o case insensitive não foi ativado na função define(). A segunda constante, “CARRO”, foi definida com o valor “Corsa” e o case insensitive foi configurado com True, isto é, aceita letras maiúsculas e minúsculas para a constante “CARRO”. A impressão da primeira constante é realizada com o valor “carro” (em minúscula). Como o parâmetro do case insensitive foi definido como True, “carro” ou “Carro” ou “CARRO” são aceitos. A impressão da constante “DADOS” só pode ser utilizada com todas as letras em miúscula, caso contrário não irá funcionar.

Figura 1 – Constantes no PHP

```
<?php

define ("DADOS", 10);

define ("CARRO","Corsa",True);

echo "Carro = " . carro . "<br>"; //ou Carro ou CARRO

echo "Dados = " . DADOS . "<br>";

echo "DADOS = " . Dados . "<br>"; // Não funciona

?>
```

Fonte: o autor.

Segundo Soares (2013), para se saber se uma constante já foi definida, utiliza-se a função **defined()**, que possui a seguinte sintaxe:

- bool defined(<nome_constante>).

A função retorna TRUE (verdadeiro) no caso de a constante estar definida. Na Figura 2 é definido uma constante chamada “DADOS” com o valor 100. Uma variável **d** é definida recebendo a constante “DADOS”. A função defined ($d) vai retornar TRUE se a constante “DADOS” atribuída à variável **d** já estiver sido definida. Como a constante “DADOS” já foi definida, será impresso que a “Constante $d está definida”.

Figura 2 – Função defined no PHP

```
<?php

define ("DADOS", 100);

echo "Dados = " . DADOS . "<br>";

$d = "DADOS";

echo "Dados = " . constant($d);

if (defined($d)){

        echo "Constante $d está definida";

}

?>
```

Fonte: o autor

2.1 Variáveis

Variáveis, segundo Soares (2013), são declaradas no PHP através do sinal $ (cifrão) seguido de um identificador único. Para se definir variáveis e seus identificadores são utilizadas as mesmas regras para constantes:

- Iniciar por uma letra ou símbolo subscrito (_);
- Pode conter letra, números ou subscrito (_).

Segundo Saraiva e Barreto (2018), o PHP é uma linguagem de programação com tipagem fraca, pois se encarrega de definir os tipos de dados da variável. Todas as variáveis da linguagem PHP são identificadas pelo cifrão ($), que precede um nome identificador.

De acordo com Miletto e Bertagnolli (2014), os nomes das variáveis são casesensitive, isto é $variavel e $Variavel são consideradas diferentes. O PHP não necessita que as variáveis sejam declaradas antes de sua

utilização, fato que a difere de outras linguagens tradicionais. A variável será criada na primeira vez que um valor for atribuído a ela. A variável não precisa ser inicializada, mas essa prática pode ser recomendável em situações em que se deseja garantir que essa variável não foi utilizada previamente.

O nome de uma variável, de acordo com Soares (2013), é case sensitive, isto é, diferencia maiúsculas de minúsculas e, diferentemente de constantes, não é possível usar o case insensitive. A Figura 3 apresenta um exemplo de declaração de duas variáveis $var, com o conteúdo "Thiago" e $VAR, com o conteúdo "Alves". Como o uso de variáveis no PHP é case sensitive, ele trata cada uma das duas variáveis de forma diferente (não considera que são a mesma variável). A impressão das duas variáveis **var**, através do comando echo, resulta em Thiago Alves.

Figura 3 – Usando variáveis no PHP

```
<?php

$var = "Thiago";

$VAR = "Alves";

echo "$var $VAR";

?>
```

Fonte: o autor

De acordo com Soares (2013), uma variável pode receber um valor diretamente, por exemplo, $valor = 10, ou então referenciar uma outra variável. Essa atribuição por referência faz com que, ao alterar o valor de uma das variáveis, ambas sejam alteradas. Para que uma variável referencie outra, precisamos apenas acrescentar o símbolo & antes do nome da variável. A Figura 4 ilustra o uso de variáveis e o processo de atribuição por valor e por referência. A variável $var recebe inicialmente a String "Maria". O valor da variável $var é atribuído diretamente (cópia)

para a variável $nome ($nome = $var). Assim, $nome recebe uma cópia da variável $var. A variável $Nome, recebe uma atribuição por referência da variável $var ($Nome = & $var). Isso quer dizer, que tanto $Nome quanto $var ocupam agora o mesmo endereço de memória. Qualquer alteração em $Nome ou $var vai refletir no mesmo conteúdo para as variáveis. A variável $nome recebe a String “Joana”. Como $nome e $var não ocupam a mesma posição de memória, $nome possui o valor “Joana” e $var continua com o valor “Maria”. A variável $Nome passa a valer “Joaquina Santos”. Como $Nome e $var ocupam a mesma posição de memória, $var, que antes possuía o valor “Maria”, passa a valer “Joaquina Santos”.

Figura 4 – Atribuições de variáveis no PHP

```
<?php

$var = "Maria";

$nome = $var; // Atribuição direta (cópia)

$Nome = &$var; // Atribuição por referência

$nome = "Joana"; //só modifica $nome

$Nome = "Joaquina Santos"; //$var é modificado

echo "\$var = $var <br>";

echo "\$nome = $nome <br>";

echo "\$Nome = $Nome <br>";

?>
```

Fonte: o autor

2.1.1 Variáveis superglobais

O PHP, de acordo com Soares (2013), possui as chamadas variáveis superglobais, que estão disponíveis em qualquer lugar do script, sem que seja necessário declará-las globais.

De acordo com Saraiva e Barreto (2018), essas variáveis superglobais estão disponíveis em qualquer local do código do programa, sem a necessidade de fazer sua inicialização ou definição.

A finalidade dessas variáveis é facilitar o acesso a dados enviados pelo servidor web (por exemplo, campos de um formulário). As seguintes variáveis superglobais estão disponíveis:

- $GLOBALS: retorna um array associativo com referência para toda e qualquer variável atualmente disponível no escopo global do script. A chave desse array é o nome das variáveis.
- $_SERVER: contém as variáveis com informações relativas ao servidor web e ao ambiente de execução do script, por exemplo:
  - DOCUMENT_ROOT: diretório raiz do script que está sendo executado.
  - PHP_SELF: nome do script em execução.
- $_GET: contém as variáveis enviadas pelo método GET. Por exemplo, se chamarmos um script da seguinte forma: http://localhost/scriptx.php?codigo=10&nome=Walace, temos que $_GET conterá dois elementos (“código” = 10, nome = “Thiago”).
- $_POST: contém as variáveis enviadas pelo método HTTP POST, por exemplo, no envio de um formulário web.
- $_COOKIE: contém as variáveis disponíveis como cookies, as quais são especiais, gravadas nas máquinas do usuário e recuperadas pelo brownser.
- $_FILES: esse array contém informações de arquivos enviados pelo computador cliente para o servidor web. Por exemplo, quando no formulário web anexar uma foto ou documento qualquer.

- $_ENV: contém as variáveis de ambiente disponíveis no momento.

- $_REQUEST: este é um array com o mesmo conteúdo de $_GET, $_POST e $_COOKIE juntos, porém não é muito confiável, uma vez que variáveis com mesmo nome serão sobrepostas conforme o valor do parâmetro *variables_order* do arquivo php.ini (o padrão é EGPCS, ou seja, Env, Get, Post, Cookie e por último Server).

- $_SESSION: contém as variáveis registradas na sessão corrente.

2.1.2 Escopo

Escopo, de acordo com Soares (2013), é o contexto em que uma variável está definida, ou seja, como é possível acessá-la. De modo geral, há apenas o escopo do script, ou seja, a variável está disponível em qualquer parte do script, inclusive em arquivos carregados dentro do script. Isso não é válido para funções e classes. A Figura 5 ilustra o conceito de escopo de variável. A variável $valor inicializada com valor 10 e a variável $valor inicializada com valor 6, dentro da função dobro(), não são a mesma variável. A variável $valor dentro da função dobro(), só existe dentro da função, não sendo visualizada nem disponível fora da função. Para tornar uma variável global, dentro de uma função, deve-se usar a palavra reservada **global** seguida do nome da variável.

Figura 5 – Escopo de variáveis no PHP

```
<?php
 $valor = 10;
 function dobro() {
    $valor = 6;
    $valor = $valor * 2;
    echo "\$valor na função " . __FUNCTION__ . " = " . $valor;
 }
 echo "\$valor = $valor <br>";
 dobro();
 echo "<br>\$valor = $valor";
?>
```

Fonte: SOARES (2013, p. 71).

3. Operadores no PHP

No PHP, segundo Saraiva e Barreto (2018), existem três grupos de operadores. São os operadores unários, os quais manipulam um único valor, por exemplo os operadores **!** (negação) e **++** (incremento). Existem os operadores binários, que manipulam dois valores ou expressões e retornam um valor, por exemplo OR (ou) e >= (maior ou igual), sendo a maioria dos operadores no PHP desta categoria. Por fim, existe o operador ternário, o qual retorna um valor entre dois possíveis, conforme o resultado de um terceiro valor ou expressão. Os operadores são divididos nas seguintes categorias:

- Operadores Aritméticos.
- Operadores de Atribuição.
- Operadores Bitwise (manipulação de bits).
- Operadores de Comparação.
- Operadores de Incremento/Decremento.
- Operadores Lógicos.

De acordo com Miletto e Bertagnolli (2014), o operador avalia uma expressão de um ou mais valores, retornando outro valor. O operador pode ser unário, quando opera em apenas um valor (!$vazio), binário, quando opera dois valor ($situação == true) e ternário, quando opera três valores.

3.1 Operadores aritméticos

Os operadores aritméticos, segundo Saraiva e Barreto (2018), são os operadores utilizados pra as operações básicas, tais como somar, subtrair, multiplicar e dividir. A Tabela 1 apresenta os operadores

aritméticos para adição, subtração, multiplicação, divisão e módulo (que retorna o resto da divisão).

Tabela 1 – Operadores Aritméticos no PHP

Operador	Objetivo	Exemplo
+	Adição	$total+$juros
-	Subtração	$total-$desconto
*	Multiplicação	$total*5
/	Divisão	$total/3
%	Módulo	$total%2

Fonte: o autor

3.2 Operadores de atribuição

O operador de atribuição, de acordo com Saraiva e Barreto (2018), deve ser usado quando se deseja atribuir o valor da expressão que está à sua direita ao operando (geralmente uma variável) que está à sua esquerda. A Tabela 2 apresenta os operadores de atribuição (=) e demais variações como atribuição por referência (=&), atribuição e adição (+=), atribuição e subtração (-=), atribuição e multiplicação (*=), atribuição e divisão (/=), atribuição e módulo (%=) e atribuição e concatenação (.=).

Tabela 2 – Operadores de Atribuição no PHP

Operador	Objetivo	Exemplo
=	Atribuição	$resultado = $total + 7
=&	Atribuição por referência	$resultado = &$total
+=	Atribuição e adição	$resultado+=10
-=	Atribuição e subtração	$resultado -=5
*=	Atribuição e multiplicação	$resultado*=6
/=	Atribuição e divisão	$resultado/=2
%=	Atribuição e módulo	$resultado%=2
.=	Atribuição e concatenação	$texto.=$x

Fonte: o autor

3.3 Operadores Bitwise

Os operadores bitwise, segundo Saraiva e Barreto (2018), são utilizados para manipulação dos bits de inteiros ou de strings (os operadores vão manipular cada caractere da string). A Tabela 3 apresenta os operadores bitwise com os operadores E (&), Ou (|), Ou exclusivo (ˆ), Negação (˜), deslocamento à esquerda (<<) e deslocamento à direita (>>).

Tabela 3 – Operadores de Bitwise

Operador	Objetivo	Exemplo
&	E (and)	$a & $b
|	Ou (or)	$a | $b
ˆ	Ou Exclusivo (Xor)	$a ^10
˜	Negação (NOT)	˜$a
<<	Deslocamento à esquerda	$a<<2
>>	Deslocamento à direita	$a>>3

Fonte: o autor

3.4 Operadores de comparação

Os operadores de comparação, de acordo com Saraiva e Barreto (2018), comparam dois valores, retornando verdadeiro ou falso. A Tabela 4 apresenta os operadores de comparação de igualdade (==), idênticos (===), diferente (!=), diferente (<>), não idênticos (!==), menor (<), menor ou igual (<=), maior (>) e maior ou igual (>=).

Tabela 4 – Operadores de Comparação

Operador	Objetivo	Exemplo
==	Igualdade	$a == $b
===	Idênticos	$a===$b
!=	Diferente	$a!=2
<>	Diferente	$a<>2
!==	Não idênticos	$a!==$b
<	Menor	$a<$b
<=	Menor ou igual	$a<=4
>	Maior	$a>$b
>=	Maior ou igual	$a>=5

Fonte: o autor

3.5 Operadores de incremento/decremento

Os operadores de incremento/ decremento, de acordo com Saraiva e Barreto (2018), são utilizados para que sejam feitas adições e subtrações diretamente na variável informada, mas sempre realizando operações unitárias, isto é, somando 1 ou subtraindo 1 da variável. Os operadores de incremento e decremento são ++ e --. As operações de incremento e decremento podem ser realizadas de duas maneiras: pós e pré. A primeira maneira retorna o valor da variável antes de incrementá-la ou decrementá-la, colocando o operador após o nome da variável. A Figura 6 ilustra o operador de pós-incremento. A variável $a é inicializada com o valor 2. A variável $b recebe a atribuição de valor de $a seguida do operador de incremento. Como o operador é de pós-incremento, a variável $b recebe o valor anterior de $a, que é 2. A variável $a, após duas operações de pós-incremento (++) passa a valer 4.

Figura 6 – Operador de pós-incremento

```
<?php

$a = 2;

$b = $a++;

$a++;

echo "$a :: $b";

?>
```

Fonte: o autor

A segunda maneira de realizar o incremento/decremento é através do pré-incremento. O incremento/decremento é realizado primeiro e depois retorna o seu valor atualizado. Para isso, deve-se utilizar o operador antes do nome da variável. A Figura 7 ilustra o operador de pré-incremento. A variável **$a** é inicializada com o valor 5. A variável **$b**

recebe a atribuição de valor de **$a** com o operador de pré-incremento. Como o operador é de pré-incremento, a variável **$b** recebe o valor atualizado de $a, que é 6. A variável $a, após duas operações de pré-incremento (++) passa a valer 7.

Figura 7 – Operador de Pré-Incremento

```
<?php

$a = 5;

$b = ++$a;

++$a;

echo "$a :: $b";

?>
```

Fonte: o autor

3.6 Operadores lógicos

Os operadores lógicos, segundo Saraiva e Barreto (2018), realizam comparações entre expressões, retornando verdadeiro ou falso como resultado. A Figura 5 apresenta os operadores E (AND), Ou (OR), Ou exclusivo (XOR), Negação (!), E (&&) e Ou (II). A diferença entre && e AND e entre II e OR está em sua ordem de precedência.

Figura 5 – Operadores Lógicos

Operador	Objetivo	Exemplo
AND	E	($a==5) AND ($b>50)
OR	Ou	($a==5) OR ($b>50)
XOR	Ou exclusivo	($a==5) XOR ($b>50)
!	Negação	!($a==5)
&&	E	($a==5) && ($b>50)
||	Ou	($a==5) \|\| ($b>50)

Fonte: o autor

Dessa forma, você aprendeu sobre constantes, variáveis e operadores usados no PHP.

TEORIA EM PRÁTICA

Ex.: Uma empresária nacional, do ramo de moda, gostaria de criar um blog para poder divulgar seus produtos. Dado a grande concorrência do setor de moda, a empresária está tendo dificuldades para divulgar seus lançamentos aos clientes e gostaria que pudesse ser criado um blog dinâmico, onde pudesse realizar as postagens de fotos de seus produtos, incluir vídeos, permitir que os clientes comentem suas postagens, conseguindo assim atingir novos públicos e maximizar suas vendas. Hoje, a empresária conta com uma loja física e envia as promoções para seus clientes via aplicativos de mensagens instantâneas e e-mails marketing. Como podemos auxiliar a empresária na escolha das variáveis que serão utilizadas para a cadastro de seus clientes?

VERIFICAÇÃO DE LEITURA

1. É utilizado para identificar um valor que está sendo usado no PHP. Quando especificada, seu valor não pode ser alterado ou removido. Assinale a alternativa que apresenta, corretamente, ao que se refere a afirmação:

   a. Constantes.

   b. Variáveis.

c. Operador de comparação.

d. Operador de incremento.

e. Operador de decremento.

2. São utilizadas no PHP através do sinal $ (cifrão) seguido de um identificador único. Assinale a alternativa que apresenta, corretamente, ao que se refere a afirmação:

a. Constantes.

b. Variáveis.

c. Operador de comparação.

d. Operador de incremento.

e. Operador de decremento.

3. São os operadores utilizados pra as operações básicas, tais como somar, subtrair, multiplicar e dividir. Assinale a alternativa que apresente, corretamente, os tipos de operadores da afirmação:

a. Operadores Lógicos.

b. Operadores de Comparação.

c. Operadores de Incremento/Decremento.

d. Operadores Aritméticos.

e. Operadores de Bitwise

Referências bibliográficas

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

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.

Gabarito

Questão 1 – Resposta: A

Resolução: As constantes possuem identificador de um valor no PHP. Quando especificadas, as constantes e seu valores não podem ser alterados ou removidos.

Feedback de reforço: Se refere a valor que não pode ser alterado em tempo de execução.

Questão 2 – Resposta: B

Resolução: As variáveis são utilizadas no PHP através do sinal $ (cifrão) seguido de um identificador único.

Feedback de reforço: Se refere a informações que podem ser alteradas ao longo do tempo de execução.

Questão 3 – Resposta: D

Resolução: Os operadores aritméticos são utilizados para as operações básicas, tais como somar, subtrair, multiplicar e dividir.

Feedback de reforço: São os operadores matemáticos.

Estruturas de controle

Autor: Thiago Salhab Alves

Objetivos

- Aprender sobre as estruturas de controle no PHP.
- Aprender sobre as estruturas de repetição no PHP.
- Aprender sobre as estruturas Break, Continue, Return e Goto.

1. Introdução

Prezado aluno, você já parou para pensar como seria o desenvolvimento de programas em PHP se não houvessem estruturas de controle e laços de repetição? Certamente, o processo de escrita e o trabalho da lógica de programação seria muito difícil, devendo ter que reescrever muitos códigos e aumentar o número de variáveis. Para isso, o PHP trabalha com estruturas de controles e estruturas de repetição para facilitar o desenvolvimento do código.

Esta leitura irá apresentar as estruturas de controle, como o comando if e estruturas de repetição como o while, do.. while e for. Você também aprenderá sobre os comandos break, continue, return e goto. Uma boa aula e bom trabalho!

PARA SABER MAIS

Pode-se suprimir as chaves das estruturas while e do.. while caso tenhamos apenas uma instrução a ser executada dentro do loop, porém é aconselhável utilizar as chaves, pois é mais seguro.

ASSIME

O PHP não suporta em uma mesma sentença formatos diferentes para if.. elseif. O mais indicado é que você escolha uma notação e a utilize em todos os seus projetos.

2. Estruturas de controle

Segundo Soares (2013), as linguagens de programação necessitam de mecanismos para realizar o controle do fluxo do programa, auxiliando na implementação de uma lógica de programação, seja ela simples ou complexa. As estruturas de controle permitem a realização, dentro dos scripts, de determinadas tarefas, como executar um grupo de instruções no caso de uma expressão ser verdadeira ou repetir um grupo de instruções uma quantidade finita de vezes, até que uma condição seja satisfeita ou enquanto uma condição não for satisfeita.

Segundo Saraiva e Barreto (2018), as estruturas de controle de um programa em PHP permitem direcionar e repetir, por meio de expressões condicionais e de iteração, o fluxo de execução das rotinas e instruções. A partir dessas estruturas é possível criar programas criativos, robustos e confiáveis, capazes de atender qualquer demanda de desenvolvimento de sistemas.

2.1 If..elseif..else

O comando **if**, de acordo com Soares (2013), permite que um grupo de instruções (delimitadas por chaves {}) possa ser executado conforme o resultado de uma expressão ou de múltiplas expressões, ou seja, se o resultado da avaliação da expressão for verdadeiro, o bloco será executado e, caso contrário, o fluxo seguirá em frente.

Também são utilizados alguns opcionais, segundo Soares (2013), como o **elseif** e **else**. Caso a expressão avaliada em **if** retorne falso e exista um comando **elseif**, este será avaliado e se sua expressão for verdadeira, as instruções relacionadas serão

executadas. No caso de retornar falso, um próximo comando **elseif** (se existir) será avaliado. Se todos os **elseif** retornarem falso em seu teste, o grupo de instruções relacionadas ao **else** será executado. É possível ter vários **elseif** em um programa, porém apenas um **else**. A sintaxe completa do **if** é apresentada a seguir:

```
if (expressão) {
   grupo de comandos
}
elseif (expressão) {
  grupo de comandos
}
elseif ...
else {
  grupo de comandos
}
```

A Figura 1 apresenta exemplo de funcionamento da estrutura de controle if. O if vai verificar se o conteúdo da variável **$valor** é menor ou igual a 10. Se o teste resultar em verdadeiro, será impresso “Valor menor que 10” e os testes são encerrados. No caso de o teste ser falso ($valor ser maior que 10), será executado o primeiro **elseif**. O **elseif** irá testar se o conteúdo da variável **$valor** é menor ou igual a 100. Se o teste resultar em verdadeiro, será impresso “Valor entre 10 e 150”. Caso o teste seja falso ($valor maior que 150), o próximo elseif será testado. O elseif irá testar se o conteúdo da variável $valor é menor ou igual a 1500. Se o teste resultar em verdadeiro, será impresso “Valor maior que 150 e menor ou igual a 1500”. Caso o teste seja falso ($valor maior que 1500) e, como não há mais comandos **elseif**, o comando **else** será executado, imprimindo “Valor acima de 1500”.

Figura 1 – Estrutura de Controle If

```
<?php

$valor = 100;

if($valor<=10) {

echo "Valor menor que 10";

}

else if($valor >10 && $valor<=150) {

echo "Valor entre 10 e 150";

}

else if($valor>150 && $valor<=1500) {

echo "Valor maior que 150 e menor ou igual a 1500";

} else

echo "Valor acima de 1000";

?>
```

Fonte: o autor.

2.2 While e do.. while

As estruturas **while** e **do.. while**, de acordo com Soares (2013), executam um grupo de comandos, repetidas vezes, enquanto uma determinada condição for verdadeira, ou seja, a cada interação do loop a expressão fornecida é avaliada e caso seja verdadeira, um novo loop é executado, caso contrário (avaliado como falso), a repetição é interrompida. A sintaxe do **while** e **do.. while** é apresentada a seguir:

```
while (expressão) {
  grupo de comandos
}

do {
  grupo de comandos
} while (expressão);
```

A diferença entre o **while** e o **do.. while**, de acordo com Soares (2013), é que o grupo de instruções do **while** pode ser executado nenhuma ou muitas vezes, pois se no primeiro teste da expressão o resultado por falso, o loop não será iniciado. Na estrutura **do.. while** o grupo de instruções será executado pelo menos uma vez, pois a avaliação da expressão só é realizada no final do loop. Assim, o comando while é conhecido como laço de repetição com teste no início e o do.. while como laço de repetição com teste no final.

A Figura 2 apresenta exemplo de funcionamento das estruturas **while** e **do.. while**. Inicialmente a variável **$i** é inicializada com valor 10. O primeiro while irá realizar a repetição enquanto **$i** for maior que 0. O laço de repetição será realizado dez vezes (de 10 a 1), pois dentro do laço há o operador de decremento vinculado à variável **$i ($i--)**. O primeiro do.. while utiliza a variável **$k** inicializada com o valor 10. O teste é realizado no final e o loop é repetido por dez vezes (de 10 a 1). No caso do segundo while, a condição de entrada no laço de repetição é enquanto **$i** for maior que 0. Como a variável **$i** está valendo zero, o laço não é executado e nenhuma iteração é realizada. Para o segundo do.. while, como o teste é realizado no final, pelo menos uma vez o laço é realizado, imprimindo o valor de **$k** que é zero.

Figura 2 – Estrutura de Repetição while e do.. while

```
<?php
 $i=10;
 echo "Primeiro while: ";
 while($i>0) {
    echo "$i ... ";
    $i--;
 }
 $k=10;
 echo "<br>Primeiro do..while: ";
 do {
    echo "$k ... ";
    $k--;
 } while($k>0);
 echo "<br>Segundo while (não teremos nenhuma iteração): ";
 while($i>0) {
    echo "$i / ";
    $i--;
 }
 echo "<br>Segundo do..while (1 iteração): ";
 do {
    echo "$k ... ";
    $k--;
 } while($k>0);
?>
```

Fonte: SOARES (2013, p. 87).

2.3 For

A estrutura de repetição **for**, segundo Soares (2013), realiza repetições de estruturas complexas, facilitando o trabalho do desenvolvedor PHP para criar lógicas mais sofisticadas e complexas. Sua sintaxe é apresentada a seguir:

```
for (expressão_1; expressão_2; expressão_3) {
  grupo de comandos
}
```

O comando **for** possui os seguintes elementos:

- expressão_1: é avaliada apenas uma vez, na primeira iteração, sendo geralmente utilizada para inicializar a variável que irá controlar o laço. Por exemplo: $i = 0;

- expressão_2: é avaliada no início de cada iteração do loop e, caso retorne falso, o **for** é encerrado. A expressão_2 é considerada expressão condicional do loop. Por exemplo: $i <= 10;

- expressão_3: é avaliada ao final de cada iteração do loop, sendo utilizada para alterar a variável de controle do laço. Por exemplo: $i ++.

A principal diferença de utilizar o while/do while e for é que os laços de repetição while e do while são utilizados para um número indefinido de repetições que serão realizados, e o laço for é utilizado para se repetir um número definidos de vezes.

A Figura 3 apresenta exemplo de funcionamento do laço de repetição **for**. O primeiro laço **for** inicializa a variável de controle $i com valor 0. A condição de repetição é enquanto o $i for menor ou igual a 15. A expressão $i++ faz o incremento da variável $i em uma unidade a cada iteração. Dessa forma, o laço será executado dezesseis vezes (de 0 a 15). O segundo laço **for** inicializa com uma quebra de linha (através do
) e a condição de repetição é quando $i for maior ou igual a zero. A expressão $i—faz o decremento da variável de controle. Dessa forma, o laço será executado dezesseis vezes (de 15 a 0).

Figura 3 – Estrutura de Repetição for

```
<?php

for($i=0;$i<=15;$i++) {

        echo "$i ... ";

}

for(print "<br>";$i>=0;$i--) {

        echo "$i ... ";

}

?>
```

Fonte: o autor.

2.4 Break

O comando **break**, de acordo com Soares (2013), afeta a execução dos comandos for, while, do.. while, permitindo que possam ser encerrados em qualquer lugar do grupo de instruções, podendo avaliar uma expressão e conforme o resultado, encerrar o a estrutura de repetição. A Figura 4 apresenta o uso do comando for. É utilizado um laço infinito, porém quando a variável $i for maior que cinquenta, o laço será encerrado.

Figura 4 – Uso do comando break

```
for(;;) {
        if($i>50){
                break;
        }
        echo "$i ... ";
        $i++;
}
```

Fonte: o autor.

2.5 Continue

A instrução **continue**, segundo Soares (2013), permite que a execução da estrutura de repetição seja alterada, mas diferentemente de break, não encerramos o loop, apenas informamos ao PHP para encerrar a iteração atual e iniciar a próxima. A Figura 5 ilustra o funcionamento do continue. A variável de controle $i inicia com valor 0. Enquanto $i for menor que 5, os valores de $i serão impressos. Se $i for igual a 2, há um comando **continue**, fazendo com que o loop inicie a próxima iteração. Assim, serão impressos os valores 0, 1, 3 e 4 (o valor 2 não será impresso).

Figura 5 – Uso do comando continue

```
<?php
for ($i = 0; $i < 5; ++$i) {
    if ($i == 2)
        continue
    print "$i\n";
}
?>
```

Fonte: o autor.

2.6 Return

O **return**, segundo Soares (2013), pode ser usado dentro de funções do PHP e seu efeito é encerrar a função atual ou o programa. O valor informado (ou expressão) na instrução **return** é retornado para o chamador. A Figura 6 ilustra o comando return. A função soma recebe por parâmetro a variável $valor e retorna $valor somado a 5. Se $valor for igual a zero, retorna -1.

Figura 6 – Uso do comando return

```
<?php

function soma ($valor) {

return $valor + 5;

}

if ($valor==0) {

return -1;

}

?>
```

Fonte: o autor.

2.7 Goto

O comando **goto**, segundo Soares (2013), é utilizado para alterar o fluxo de execução do processo, desviando de uma seção do programa para outra. Este comando só passou a estar disponível após a versão 5.3 do PHP, porém recomenda-se a sua não utilização, pois apresenta alguns inconvenientes, sendo o principal deles o de tornar o código muito difícil de ser entendido, principalmente se utilizar um grande número de goto no código. A Figura 7 ilustra o funcionamento do comando goto. Se o valor da variável $i for igual a 49, o comando goto fará um deslocamento para uma nova seção do código, chamada de final. Nesta seção fim, será impresso que “i é igual a 49”.

Figura 7 – Uso do comando goto

```
<?php

for($i=0; $i<50; $i++) {
if($i == 49) goto final;
}
echo “i = $i”;
final:
echo “i é igual a 49”;
?>
```

Fonte: o autor.

Dessa forma, você aprendeu sobre as estruturas de controle como o comando if e estruturas de repetição como o while, do.. while e for. Você também aprendeu sobre os comandos break, continue, return e goto.

TEORIA EM PRÁTICA

Ex.: Um programador está desenvolvendo uma interface para cadastro de clientes em que se deve preencher os dados e se deparou com a necessidade de utilizar laços de

repetição. Ao procurar na literatura sobre o uso de laços de repetição no PHP, observou que os laços **while**, **do.. while** e **for** são os utilizados pelos programadores. Porém o programador ficou na dúvida de quando utilizar cada tipo de laço de programação e se o resultado seria o mesmo. Como podemos auxiliar o programador a determinar quando utilizar cada tipo de laço de repetição para um melhor aproveitamento do recurso?

VERIFICAÇÃO DE LEITURA

1. Os laços de repetição executam um grupo de comandos, repetidas vezes, enquanto uma determinada condição for verdadeira, ou seja, a cada interação do loop a expressão fornecida é avaliada, e caso seja verdadeira, um novo loop é executado, caso contrário (avaliado como falso), a repetição é interrompida. Assinale a alternativa que apresenta, corretamente, o tipo de comando de repetição que realiza teste no final:

   a. Apenas while.

   b. Apenas do.. while.

   c. Apenas for.

   d. While e do.. while.

   e. While, do.. while e for.

2. O comando for realiza repetições de estruturas complexas, facilitando o trabalho do desenvolvedor PHP para criar lógicas mais sofisticadas e complexas.

Assinale a alternativa que apresenta, corretamente, a quantidade de expressões que compõem o laço for:

a. Uma expressão.

b. Duas expressões.

c. Três expressões.

d. Quatro expressões.

e. Cinco expressões.

3. “É utilizado para alterar o fluxo de execução do processo, desviando de uma seção do programa para outra”. Assinale a alternativa que apresente, corretamente, o tipo de comando que a afirmação apresenta:

a. Break.

b. Continue.

c. Goto.

d. Return.

e. Else.

Referências bibligráficas

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

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.

Gabarito

Questão 1 – Resposta: B

O comando de repetição que realiza teste no final é o do.. while.

Feedback de reforço: Inicia executando os comandos sem realizar nenhum teste inicialmente.

Questão 2 – Resposta: C

O comando de repetição for é usado uma quantidade finita de vezes.

Feedback de reforço: Expressões de inicialização, verificação e incremento/decremento compõem o laço for.

Questão 3 – Resposta: C

O comando goto é utilizado para alterar o fluxo de execução do processo, desviando de uma seção do programa para outra.

Feedback de reforço: O comando se refere ao deslocamento do código para outra seção.

Funções

Autor: Thiago Salhab Alves

Objetivos

- Aprender a criar e utilizar funções no PHP;
- Aprender sobre argumentos e retorno de valores de uma função;
- Aprender sobre funções recursivas e funções anônimas.

1. Funções

Prezado aluno, você já parou para pensar como seria o desenvolvimento de programas em PHP se não houvessem as funções? Certamente, o processo de escrita e o trabalho da lógica de programação seria muito difícil, devendo repetir os códigos para uma tarefa. Para isso, o PHP trabalha com funções, que é subprograma que executa uma série de instruções e pode retornar ou não um valor como resultado.

Esta leitura irá apresentar como criar e utilizar funções no PHP. Você também aprenderá sobre argumentos, retorno de valores de uma função, variáveis funções, funções recursivas e funções anônimas. Uma boa aula e bom trabalho!

PARA SABER MAIS

Uma função pode estar em qualquer lugar do script, inclusive após a sua chamada, ou seja, não é necessário especificar uma função antes de chamá-la. Qualquer instrução PHP pode estar em uma função, inclusive em outras funções e classes.

ASSIMILE

O valor padrão de um argumento deve ser literal (um número, uma string) ou uma constante, ou seja, não podemos utilizar variáveis, classes ou funções como valor padrão de um argumento.

1.1 Criando e utilizando funções

Segundo Soares (2013), a criação de uma função é realizada através do uso da palavra reservada **function** juntamente com o nome da função, que deve ter seu identificador único. A seguir, pode-se incluir dentro dos parênteses a lista de argumentos da função e, para finalizar, inserir o código da função, delimitado por chaves { }. A estrutura de uma função é realizada da seguinte forma:

```
function nome_função (argumentos) {
    grupo de comandos da função
}
```

O nome da função, de acordo com Saraiva e Barreto (2018), não é *case sensitive*, ou seja, não há diferença entre minúsculas e maiúsculas (uma_função é igual à Uma_Função). Os parênteses são obrigatórios. Mesmo que não haja argumentos da função, deve-se incluir os parênteses após o nome da função. Qualquer instrução PHP pode estar em uma função, inclusive em outras funções e classes. A função pode estar em qualquer lugar do script, inclusive após sua chamada, não sendo necessário especificar uma função antes de chamá-la.

A Figura 1 apresenta um exemplo de uso de função no PHP. A função se chama **função ( )**, sendo criada pelo comando function, não recebendo nenhum parâmetro (os parênteses estão vazios) e, possui associada a ela, um laço de repetição **for**, variando a variável $i de zero a dez, imprimindo a cada iteração o quadrado do valor de $i. Para que a função possa ser chamada, utiliza-se o seu nome função ( ).

Figura 1 – Exemplo de Função em PHP

```
<?php
        echo “Trabalhando com funções”;
        função ( );
        function função ( ) {
                echo “<br>”;
                for ($i=0; $i<10; $i++) {
                        echo “O Quadrado de “ . $i . ” é  “  ” . ($i * $i);
                }
        }
?>
```

Fonte: o autor.

O PHP, segundo Soares (2013), permite que sejam criadas funções condicionais, que são aquelas em que dependendo do valor de uma variável, a função irá existir ou não. Nesses casos, é necessário primeiro criar a função para que depois possa ser chamada. A definição de uma função condicional é realizada com a instrução **if**. A Figura 2 apresenta um exemplo de função condicional. A função valida( ) somente será executada se o conteúdo da variável $status for verdadeiro (TRUE), caso contrário, a função não será executada.

Figura 2 – Exemplo de Função Condicional em PHP

```
<?php
$status = TRUE;
if ($status){
function valida(){
echo “Função validada!!”;
   }
 }
?>
```

Fonte: o autor.

As funções permitem, segundo Saraiva e Barreto (2013), receber um ou mais valores que podem ser enviados quando as funções são chamadas. Os valores que são enviados são chamados de parâmetros ou argumentos da função. Uma função pode ter nenhum ou muitos argumentos, bastando para isso informá-los na declaração da função e, se houver mais argumentos, separá-los por vírgula. A Figura 3 apresenta um exemplo de função que recebe um argumento por parâmetro. A função quadrado ( ) recebe um argumento ($valor) por parâmetro. Ao chamar a função quadrado ( ), o argumento quatro é enviado por parâmetro. Este argumento é recebido pela função e utilizado para determinar o quadrado do argumento ($valor * $valor).

Figura 3 – Exemplo de Função com Argumento no PHP

```
<?php
 function quadrado($valor) {
    echo "Quadrado de $valor é: " . ($valor*$valor)
 }
 quadrado(4);
?>
```

Fonte: SOARES (2013, p. 101).

De acordo com Saraiva e Barreto (2018), a regra padrão para utilização de argumentos de uma função é que eles sejam passados por valor, isto é, a função faz uso de uma cópia da variável que é enviada e não a variável em si. Assim, alterações que ocorrerem dentro da função não irão afetar a variável original. Na Figura 4 é apresentado um exemplo de passagem de parâmetro por valor. A variável $valor1 recebe o valor dez e $valor2 recebe o valor 20. Ao chamar a função soma ( ), os valores 10 e 20 são enviados por parâmetro para a função. Dentro da função, a variável $resultado recebe a soma dos valores, passando a valer 30.

Figura 4 – Exemplo de função com passagem de parâmetro por valor

```
<?php
function soma($valor1, $valor2){
$resultado = $valor1 + $valor2;
echo “Soma: $resultado”;
}
$valor1 = 10;
$valor2 = 20;
soma($valor1, $valor2);
?>
```

Fonte: o autor.

Para que esse comportamento possa ser alterado, segundo Soares (2013), para que o argumento possa ser passado como referência, deve-se utilizar o operador **&** antes do nome do argumento. Assim, qualquer alteração realizada na função irá refletir na variável original. Na Figura 5 é apresentado um exemplo de passagem de parâmetro por referência. A variável $valor possui o valor 12. Ao chamar a função quadrado ( ), o valor 12 é enviado por parâmetro para a função.
A função recebe $valor com a inclusão do & antes do nome da variável $valor, indicando que se refere ao endereço de memória da variável $valor. Dentro da função, a variável $valor passa a valer 144 (o valor de 12 ao quadrado). Assim, o conteúdo da variável de origem $valor, que era 12, passa a valer 144 também.

Figura 5 – Exemplo de função com passagem de parâmetro por referência

```
<?php
 function quadrado(& $valor) {
    $val_orig = $valor;
    $valor *= $val_orig;
    echo "Quadrado de $val_orig é: " . $valor;
 }
 $valor = 12;
 echo "Valor original: $valor <br>";
 quadrado($valor);
 echo "<br>Valor atual: $valor";
?>
```

Fonte: SOARES (2013, p. 102).

Há algumas situações, segundo Saraiva e Barreto (2018), em que é necessário definir um valor padrão para alguns ou todos os argumentos de uma função. Isso é realizado acrescentando, após o argumento, o operador de atribuição (=) e o valor que se deseja como padrão. A Figura 6 apresenta um exemplo de uso de função com valor padrão. A função padrão possui os argumentos $valor1 = “HTML” e $valor2 = “PHP”. Assim, quando a função valores é chamada sem parâmetro, valores ( ), a mensagem a ser exibida será “O valor1 é HTML e o valor2 é PHP”. Ao chamar a função valores (“CSS”, “C++”) será exibida a mensagem “O valor1 é CSS e o valor2 é C++”. Porém, ao enviar apenas um parâmetro, valores(“Java”), apenas o primeiro parâmetro é afetado, produzindo a mensagem “O valor1 é Java e o valor2 é PHP”.

Figura 6 – Exemplo de função com passagem de parâmetro com valor padrão

```
<?php

function valores($valor1="HTML", $valor2="PHP"){

        echo "O valor1 é $valor1 e o valor2 é $valor2"."<br>";

}

valores("Java");

valores();

valores("CSS", "C++");



?>
```

Fonte: o autor.

É possível também, de acordo com Saraiva e Barreto (2018), fazer com que uma função retorne um valor, que pode ser qualquer tipo aceito pelo PHP (inteiro, ponto flutuante, booleano, String, array, etc.). Para que isso seja possível, deve-se utilizar a instrução **return** seguida do valor

que se deseja retornar. O exemplo da figura 7 ilustra a função média, que recebe por parâmetro dois valores, 9 e 8. A função calcula a média e armazena seu resultado em $m, retornando a média calculada (8,5) para o ponto de chamada.

Figura 7 – Exemplo de função com retorno de valores

```
<?php
function media($valor1, $valor2) {
   $m = ($valor1+$valor2)/2;
   return $m;
}
$valor1 = 9;
$valor2 = 8;
echo “A média dos valores é ” . media($valor1, $valor2);
?>
```

Fonte: o autor.

Da mesma forma que há argumentos enviados por referência, pode-se ter funções que retornam valores por referência. Para que isso ocorra, deve-se incluir o operador **&** antes do nome da função e também na sua chamada. A Figura 8 apresenta exemplo de função com retorno de valores por referência. A função soma ( ) é chamada utilizando o **&**, indicando que se trata de retorno de valor por referência. A função soma também apresenta o **&** em sua estrutura.

Figura 8 – Exemplo de função com retorno de valores por referência

```
<?php
function &soma($valor1, $valor2) {
$resultado = $valor1+$valor2;
return $resultado;
}
$res = &soma(10,20);
?>
```

Fonte: o autor.

O PHP também permite, segundo Soares (2013), utilizar recursividade de funções, que é a possibilidade de uma função chamar a ela mesma. Assim, pode-se realizar cálculos complexos de modo simples e prático. Para o correto funcionamento da estrutura recursiva é necessário a definição de um ponto de parada, caso contrário, haverá um loop infinito. A Figura 9 ilustra uma função recursiva fatorial ( ). Dentro do corpo da função fatorial ( ), há uma chamada à própria função fatorial ( ), sendo uma chamada recursiva. O ponto de parada das chamadas é quando $n for igual a zero. A recursividade faz uso da estrutura de dados pilha, realocando todas as variáveis envolvidas no processo recursivo.

Figura 9 – Exemplo de função recursiva

```
<?php
 function fatorial($n) {
    if($n<0) {
       return "Não existe fatorial de número negativo";
    }
    elseif($n<=1) {
       return 1;
    }
    else {
       return $n * fatorial($n-1);
    }
 }
 $v = 10;
 echo "Fatorial de $v = " . fatorial($v);
?>
```

Fonte: SOARES (2013, p. 106).

Um conceito utilizado no PHP, segundo Soares (2013), são as funções anônimas, que são funções criadas dentro de uma variável do PHP, chamadas funções lambda, permitindo criar funções dinâmicas dentro do PHP. Para se criar funções anônimas, deve-se utilizar a instrução **create_function** que possui a seguinte sintaxe:

string create_function(Argumentos, lógica_da_função)

Os argumentos são informados em uma string delimitada por aspas simples, assim como o código, pois isso impede qualquer problema de

interpretação do que se está enviando à função, já que as referências às variáveis (no argumento e na lógica) devem ser feitas usando estruturas do PHP, as variáveis começando pelo símbolo **$**. A codificação da lógica deve seguir as regras da linguagem. A Figura 10 apresenta a criação de uma função anônima através do comando create_function, recebendo dois parâmetros, $valor e $exp e retornando o $valor elevado a $exp, utilizando para isso a função pow.

Figura 10 – Exemplo de função anônima

```
<?php
 // Cria a função
 $func = create_function('$valor,$exp','return "$valor elevado a $exp é igual
                          à " . pow($valor,$exp) . "<br>";');
 echo "Nome da função: $func <br>";
 echo $func(13,5);
 echo $func(5,7);
 echo $func(2,10);
?>
```

Fonte: SOARES (2013, p. 107).

Dessa forma, você aprendeu sobre criar e utilizar funções no PHP. Você também aprendeu sobre argumentos, retorno de valores de uma função, variáveis funções, funções recursivas e funções anônimas.

TEORIA EM PRÁTICA

Você é um programador PHP e está escrevendo seu novo código para um projeto e percebeu a necessidade de utilizar funções para o cadastro e exibição dos dados de clientes. Ao procurar na literatura sobre o uso de funções, se deparou com funções sem passagem de parâmetro e funções com passagem de parâmetro por valor e referência. Porém, ficou na dúvida de quais funções criar para poder cadastrar e exibir os dados dos clientes. Como determinar quais funções devem ser utilizadas para o cadastro e exibição dos dados dos clientes?

VERIFICAÇÃO DE LEITURA

1. Qualquer instrução PHP pode estar em uma função, inclusive em outras funções e classes. A função pode estar em qualquer lugar do script, inclusive após sua chamada, não sendo necessário especificar uma função antes de chamá-la. A criação de uma função é realizada através do uso de uma palavra reservada. Assinale a alternativa que apresenta, corretamente, qual palavra reservada se utiliza para criação de funções:

   a. Function.

   b. Procedure.

   c. Void.

   d. Funct.

   e. Create function.

2. As funções permitem receber um ou mais valores que podem ser enviados quando as funções são chamadas. Assinale a alternativa que apresenta, corretamente, o termo usado para o recebimento desses valores:

   a. Valores.

   b. Argumentos ou parâmetros.

   c. Referências.

   d. Expressões.

   e. Dados.

3. O PHP também permite a possibilidade de uma função chamar a ela mesma. Assim, pode-se realizar cálculos complexos de modo simples e prático. Assinale a alternativa que apresente, corretamente, este tipo de recurso:

   a. Passagem de parâmetro por valor.

   b. Passagem de parâmetro por referência.

   c. Recursividade.

   d. Retorno.

   e. Repetição.

Referências bibliográficas

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

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.

Gabarito

Questão 1 – Resposta: A

Resolução: A criação de uma função é realizada através do uso de uma palavra reservada function.

Feedback de reforço: Comando que representa função.

Questão 2 – Resposta: B

Resolução: As funções permitem receber um ou mais valores que podem ser enviados quando as funções são chamadas. Esses valores são chamados de argumentos ou parâmetros.

Feedback de reforço: Lista de valores que a função pode receber.

Questão 3 – Resposta: C

Resolução: O PHP também permite a possibilidade de uma função chamar a ela mesma. Este recurso é chamado de recursividade.

Feedback de reforço: Refere-se ao conceito de recursão utilizado na matemática.

PHP e orientação a objetos

Autor: Thiago Salhab Alves

Objetivos

- Aprender os conceitos de Orientação a Objetos;
- Aprender a utilizar a Orientação a Objetos no PHP;
- Aprender sobre controle de exceções no PHP.

1. PHP e orientação a objetos

Prezado aluno, você já parou para pensar como a programação em PHP seria beneficiada com o uso da Orientação a Objetos? Certamente, o processo de escrita do código e a lógica envolvida no desenvolvimento deste seriam facilitados devido à organização em termos de classes e com os recursos de herança, o que permitiria também o reaproveitamento de inúmeros códigos já construídos, evitando que necessitassem ser reescritos. O PHP, à partir de sua versão 5, permite a escrita de códigos orientados a objetos.

Esta leitura irá apresentar os conceitos de orientação a objetos. Você também aprenderá a utilizar a orientação a objetos no PHP e sobre controle de exceções. Uma boa aula e bom trabalho!

PARA SABER MAIS

Uma das mais importantes mudanças no PHP, e provavelmente a principal, foi a total remodelação da linguagem para uma melhor aderência ao conceito de OO (Orientação a Objetos). Até a versão 4 o PHP tratava os objetos como tipos primitivos (da mesma forma que trata inteiros ou strings, por exemplo). Já na versão 5 do PHP a manipulação de objetos foi totalmente reescrita, permitindo uma melhor performance e muitas funcionalidades adicionais.

ASSIMILE

Classes que estendem outras classes, ou seja, subclasses, não executam automaticamente o construtor da superclasse. O método precisa ser invocado

explicitamente através de parent::__construct().
O mesmo vale para a função __destruct().

1.1 Orientação a objetos

A orientação a objetos também denotada por OO consiste, segundo Soares (2013), no desenvolvimento de uma estratégia em que os sistemas devem ser construídos baseados em uma coleção de componentes reusáveis conhecidos como objetos, cujas características fundamentais são:

- Objetos possuem dados;
- Objetos manipulam dados e fazem coisas.

Considere como exemplo uma empresa que possui basicamente funcionários e departamentos. Sabe-se que funcionários possuem dados, tais como nome, função, número do crachá, quem é seu chefe, salário, departamento ao qual pertencem, etc., e também realizam operações (executam suas funções, preparam relatórios, marcam ponto, saem de férias, etc.). Há também que departamentos, do ponto de vista sistêmico, sabem coisas (seu código, seu nome) e fazem coisas (incluir funcionários no departamento).

Assim, do ponto de vista da orientação a objetos (OO), terão duas classes (uma é a representação genérica de um objeto, por exemplo, João, Maria e Pedro, que são pessoas no mundo real, mas que podem ser modeladas na classe funcionário), uma classe chamada funcionário e a classe departamento.

1.1.1 Objeto

Segundo Saraiva e Barreto (2013), objeto consiste na entidade que se deseja generalizar (torná-la uma classe) e pode ser, por exemplo,

uma pessoa, um lugar, um evento, um conceito, um relatório, uma tela, enfim qualquer coisa real. Pensando em termos de um banco de dados, pode-se enxergar um objeto como uma das linhas de uma tabela (por exemplo, um registro da tabela de clientes, de produtos, ou mesmo de condições de pagamento), mas lembre-se, no entanto, em orientação a objetos (OO), este objeto deve ter, além de dados, funcionalidades relacionadas a ele.

1.1.2 Classe

Saraiva e Barreto (2018) explicam a classe como a abstração de um objeto, ou seja, é um modelo a partir do qual os objetos podem ser criados. Enquanto um objeto representa algo que existe no mundo real, uma classe é a generalização deste, mostrando quais devem ser suas características (dados) e quais as suas capacidades (funcionalidades). Tome, por exemplo, a classe funcionário. Ela deve descrever a entidade funcionário a qual deve ter características próprias, tais como nome, código, função, salário e deve ter capacidades (funcionalidades). Mas, em termos de objeto não se pensaria na classe funcionário, mas em um funcionário específico, por exemplo, o funcionário Pedro, o qual possui um nome (Pedro), um código (12345), uma função (Analista de sistemas), um salário (R$ 3000,00/mês).

1.1.3 Atributo

Atributo consiste em 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 (SOARES, 2013).

1.1.4 Método

De acordo com Saraiva e Barreto (2018), os métodos definem o que as classes sabem fazer, ou seja, os métodos podem alterar os atributos e

realizar funções inerentes à classe (como 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.

O código apresentando na Figura 1 apresenta a criação da classe Aluno em linguagem PHP. A classe apresenta como atributos as variáveis $ra, $nome e $curso e os métodos getCurso() e setCurso(), que retornam e armazenam os cursos do aluno, respectivamente.

Figura 1 – Exemplo de classe em PHP

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

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

Fonte: o autor.

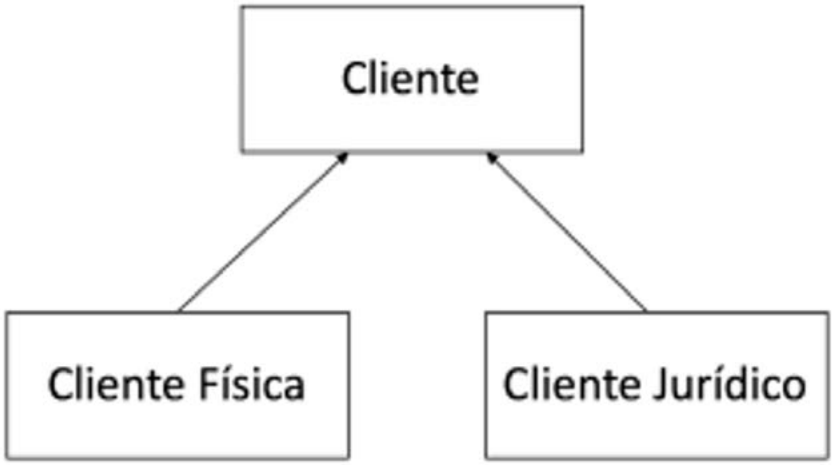
1.1.5 Herança

No desenvolvimento de sistemas, dos mais simples aos mais complexos, frequentemente são construídas classes que muitas vezes são similares a outras já existentes, as quais compartilham parte de suas definições com atributos e métodos. O uso da herança é utilizado para reaproveitar os atributos e métodos de uma classe em outra, alterando somente o que é particular da nova classe (SARAIVA; BARRETO, 2018).

Herança, de acordo com Soares (2013), 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). Considere por exemplo as entidades cliente pessoa física e cliente pessoa jurídica, que possuem os atributos código, nome, endereço, telefone, contato e alguns métodos em comum.

Assim, pode-se criar uma classe genérica chamada cliente e fazer com que as classes cliente pessoa física e cliente pessoa jurídica herdem os atributos e métodos da classe cliente, realizando as alterações inerentes a cada uma, como CPF na classe cliente pessoa física e CNPJ na classe cliente pessoa jurídica. A grande vantagem de utilização de herança está na reusabilidade dos códigos escritos pelos programadores, pois sem esta possibilidade seria necessário alterar um mesmo item em vários pontos distintos do sistema. O diagrama mostrado na Figura 2 ilustra o exemplo de herança, com a classe pai ou genérica Cliente e as classes específicas Cliente Física e Cliente Jurídico.

Figura 2 – Exemplo de Herança



Fonte: o autor

1.2 Utilização de orientação a objetos no PHP

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.

1.2.1 Classe

Uma classe no PHP, segundo Saraiva e Barreto (2018), deve ser definida da seguinte forma:

```
class nome_classe {
        Atributos
        Métodos
}
```

O código escrito em linguagem PHP, mostrado na Figura 3 , ilustra a criação de uma classe carro. Esta classe possui os atributos $_marca, $_modelo, $_cor e $_ano, os métodos para atribuir os valores para os atributos setMarca(), setModelo(), setcor(), setAno() e os métodos para obter os valores dos atributos getMarca(), getModelo(), getCor(), getAno() e getCarro(). Por padrão, utiliza-se os métodos com nomenclatura iniciada com **set** para atribuição de valores para os atributos e os métodos com nomenclatura iniciada com **get** para recuperar os valores dos atributos. Dentro dos métodos da classe foram utilizados a variável **$this** para referenciar a própria classe e este deve ser o procedimento para referenciar a própria classe em seus métodos. Para se instanciar (criar um objeto a partir de uma classe existente) uma classe no PHP, deve-se utilizar a instrução **new** seguida do nome da classe e, se for necessário, os parâmetros exigidos no método de construção da classe:

```
<?php
        $_c = new carro();
?>
```

Figura 3 – Exemplo de classe no PHP

```php
<?php
    class carro {
        private $_marca;
        private $_modelo;
        private $_cor;
        private $_ano;

        public function setMarca($_m) {
            $this->_marca = $_m;
        }

        public function setModelo($_m) {
            $this->_modelo = $_m;
        }

        public function setcor($_c) {
            $this->_cor = $_c;
        }
        public function setAno($_a) {
            if(is_int($_a)) {
                $this->_ano = $_a;
            }
            else {
                return FALSE;
        }
        public function getMarca() {
            return $this->_marca;
        }

        public function getModelo() {
            return $this->_modelo;
        }

        public function getCor() {
            return $this->_cor;
        }

        public function getAno() {
            return $this->_ano;
        }

        public function getCarro() {
            return    "Marca: " . $this->getMarca() .  "<br/>" .
                        "Modelo: " . $this->getModelo() . "<br/>" .
```

Fonte: Soares (2013, p. 294).

Uma vez que a classe é instanciada, pode-se acessar os atributos e métodos públicos da classe. Deve-se utilizar o operador -> após o nome do objeto que instancia a classe. O código apresentado na Figura 4 ilustra o exemplo de instanciação da classe carro e a atribuição da marca “Toyota” através do método setMarca(), modelo “Corolla” através do método setModelo(), da cor “Preto” através do método setcor() e ano 2019 através do método setAno(). Através da chamada do método getCarro(), exibe-se a marca, modelo, cor e ano do veículo cadastrado.

Figura 4 – 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: o autor.

1.3 Visibilidade de atributos e métodos

Os métodos e atributos de uma classe no PHP, segundo Saraiva e Barreto (2018), podem ser definidos como privativos (private), públicos (public) e protegidos (protected). Cada um dos tipos definem como o atributo ou método se comporta perante a classe, subclasses e o restante do sistema.

1.3.1 Public

Um atributo ou método definido como public (público) torna-o acessível em qualquer lugar da classe, de suas subclasses, bem como em qualquer parte dos scripts que contêm a classe. O código apresentado na Figura 5 mostra um exemplo de uso de atributo público na linguagem PHP. O atributo $ra foi definido como público assim como o método setRa(). Assim, é possível acessar o atributo tanto dentro quanto fora da classe.

Figura 5 – 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: o autor.

1.3.2 Protected

Atributos ou métodos definidos como protected (protegido) são visíveis pela classe que os criou e por suas subclasses (classes que herdam a classe principal), porém não são acessíveis fora deste contexto (fora da classe principal ou suas subclasses, por exemplo, no script que contém a definição da classe). O código apresentado na Figura 6 apresenta um exemplo de uso de atributo protegido. O atributo $ra foi definido como protegido assim como $tcc.

Figura 6 – 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: o autor.

1.3.3 Private

Os atributos e os métodos definidos como private (privativo) são visíveis apenas na classe que os criou, ou seja, subclasses ou o script que contêm a classe não podem acessar esses atributos ou métodos. O código apresentado na Figura 7, o atributo $_tipo foi definido como privado. Dessa forma, o acesso direto não é permitido. Para que o $_tipo seja modificado, deve ser utilizado o método setTipo().

Figura 7 – Exemplo de atributo privado no PHP

```
<?php
class pessoas {
        private $_tipo;
        protected $_nome;
        protected $_endereço;
        protected $_telefone;
        protected function setTipo($_t) {
            $this->_tipo = $_t;
        }
}
class estudante extends pessoas {
        protected $_curso;
        function __CONSTRUCTOR(){
                parent::setTipo("E");
        }
}
$e = new estudande();
$e->_tipo = "A"; // Erro
?>
```

Fonte: o autor.

1.4 Construtor e destruidor de classes

O construtor, 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 através da função __destruct(), é executado sempre que o objeto for destruído, seja explícita ou implicitamente (por exemplo, no término do script que utilizava a classe). A função __construct() pode ser construída de forma que aceite parâmetros na sua definição, já a função __destruct() não aceita parâmetros (SARAIVA; BARRETO, 2018).

1.5 Controle de exceções

É possível criar blocos do tipo try/catch, comuns em outras linguagens orientadas a objeto (tais como Java e Object Pascal) e existe ainda a classe Exception que disponibiliza melhor controle das exceções geradas nos scripts. As funções de manipulação de erros disponíveis nas versões anteriores da linguagem PHP continuam válidas e mantendo sua utilidade, porém nas versões mais recentes da linguagem PHP há um maior número de ferramentas disponíveis para um gerenciamento mais limpo e organizado dos erros e exceções. A classe Exception possui os seguintes métodos:

- __construct: no construtor da classe é obrigatória a definição de uma mensagem e opcionalmente o número de erro associado.
- getMessage: retorna a mensagem de erro.
- getCode: o código do erro.
- getFile: nome do arquivo no qual a exceção foi gerada.
- getLine: número da linha em que a exceção foi gerada.
- getTrace: um array com informações sobre cada etapa na progressão de uma exceção.
- getTraceAsString: o mesmo que getTrace(), só que em uma string em vez de um array.

O código apresentado na Figura 8 ilustra o uso de controle de exceção na linguagem PHP. A classe exception é muito útil quando utilizada em conjunto com a palavra reservada thrown, a qual encerra o método executado imediatamente e lança a exceção gerada para o sistema. Uma outra forma de gerenciamento de exceções é a utilização da estrutura try/catch/finally. Essa estrutura consiste em uma cláusula try seguida de um ou mais blocos catch e, opcionalmente, de um único bloco finally.

Figura 8 – Exemplo de controle de exceção no PHP

```php
<?php
    class mostra_arquivo {
        protected $_nome_arq;
        private $_flg;

        function __construct() {
            $this->_nome_arq = $_nome;
            $this->_flg = FALSE;
        }

        public function setArq($_nome) {
            if(file_exists($_nome)) {
                $this->_nome_arq = $_nome;
                $this->_flg = TRUE;
            }
            else {
                throw new exception("Arquivo Informado não Existe",4);
            }
        }

        public function mostra() {
            if($this->_flg===FALSE) {
                throw new exception("Arquivo não informado",10);
            }
            elseif(file_exists($this->_nome_arq)) {
                if(filesize($this->_nome_arq)==0) {
                    throw new exception("Arquivo Vazio");
                }
                echo "<pre>";
                echo htmlentities(file_get_contents($this->_nome_arq));
                echo "</pre>";
            }
            else {
                throw new exception("Arquivo Informado não Existe",40);
            }
        }
    }
    function exibe($_na) {
        $_a = new mostra_arquivo;
        try {
            $_a->setArq($_na);
            $_a->mostra();
        } catch (Exception $e) {
            echo "ERRO: " . $e->getCode() . "-" . $e->getMessage();
            if($e->getCode()>10) {
                exit();
            }
        }
    }
    exibe("Arquivo.txt"); // Erro
    exibe("class_11.php5");  // OK
?>
```

Fonte: Soares (2013, p. 315).

Dessa forma, você aprendeu os conceitos de orientação a objetos. Você também aprendeu a utilizar a orientação a objetos no PHP e aprendeu sobre controle de exceções.

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?

VERIFICAÇÃO DE LEITURA

1. A orientação a objetos (OO) é o desenvolvimento de uma estratégia em que os sistemas devem ser construídos baseados em uma coleção de componentes reusáveis. Assinale a alternativa que apresenta, corretamente, quais são esses componentes reusáveis:

   a. Classes.

   b. Objetos.

   c. Atributos.

   d. Métodos.

   e. Construtor.

2. Definem o que as classes sabem fazer, ou seja, podem alterar os atributos e realizar funções inerentes à classe. Assinale a alternativa que apresenta, corretamente, ao que se refere a definição apresentada:

   a. Classes.

   b. Objetos.

   c. Atributos.

   d. Métodos.

   e. Herança.

3. Os métodos e atributos de uma classe no PHP podem ser definidos através de algumas visibilidades.
   Há um tipo de visibilidade que torna atributos e métodos acessível em qualquer lugar da classe, de suas subclasses, bem como em qualquer parte dos scripts que contêm a classe. Assinale a alternativa que apresente, corretamente, este tipo de visibilidade:

   a. Public.

   b. Private.

   c. Protected.

   d. Void.

   e. Function.

Referências bibliográficas

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

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.

Gabarito

Questão 1 – Resposta: B

Resolução: A orientação a objetos (OO) é o desenvolvimento de uma estratégia em que os sistemas devem ser construídos baseados em uma coleção de componentes reusáveis chamados objetos.

Feedback de reforço: São a instância de uma classe.

Questão 2 – Resposta: D

Resolução: Os métodos definem o que as classes sabem fazer, ou seja, podem alterar os atributos e realizar funções inerentes à classe.

Feedback de reforço: São as funções desempenhadas pela classe.

Questão 3 – Resposta: A

Resolução: A visibilidade public torna atributos e métodos acessíveis em qualquer lugar da classe, de suas subclasses, bem como em qualquer parte dos scripts que contêm a classe.

Feedback de reforço: Tipo de visibilidade padrão para atributos e métodos.

PHP e MySQL

Autor: Thiago Salhab Alves

Objetivos

- Aprender a instalar o MySQL;
- Aprender a criar banco de dados no MySQL;
- Aprender a utilizar o PHP com MySQL.

1. PHP e MySQL

Prezado aluno, você já parou para pensar como a programação em PHP seria beneficiada com a possibilidade da conexão com banco de dados? Certamente, o processo de escrita e o trabalho da lógica de programação seria facilitado e a criação de páginas dinâmicas otimizado. O PHP permite a conexão com banco de dados, apresentando classes para conexão e manipulação de dados.

Esta leitura irá apresentar como instalar o MySQL. Você também aprenderá a criar banco de dados no MySQL e utilizar o PHP com MySQL. Uma boa aula e bom trabalho!

PARA SABER MAIS

Para saber a situação atual do parâmetro autocommit do banco de dados MySQL, utilize o comando SQL "select @@autocommit", o qual retorna 1 se autocommit estiver habilitado e 0 caso contrário. Desligada a autoconfirmação de gravação dos registros, pode ser utilizado os métodos para efetuar (commit) ou descartar (rollback) as transações enviadas ao banco de dados.

ASSIMILE

Assim como na versão anterior do MySQL, a primeira ação que deve ser executada é a conexão do servidor MySQL e a seleção do banco de dados. No mysqli isso é realizado em uma única etapa, diferente da versão anterior que era necessário o uso de duas funções, sendo mysql_connect e mysql_select_db.

1.1 Introdução ao uso do PHP e MySQL

Segundo Soares (2013), o desenvolvimento de sites dinâmicos traz consigo a necessidade de acesso a algum tipo de banco de dados relacional. Isto acontece em alguns tipos de sites somente para algumas tarefas simples como guardar usuários e suas senhas, e em outros sites com sistemas de bancos de dados complexos, em que o site não funciona sem esses dados (uma loja virtual, por exemplo).

O PHP é uma das linguagens com maior disponibilidade de acesso a bancos de dados, pois com ele podemos acessar o Oracle, SQL Server, PostgreSQL, FireBird, MySQL, SysBase, Informix, SQLite e vários outros bancos de dados, além de ser possível utilizar drives ODBC para acesso aos bancos que não possuem um módulo específico no PHP.

Segundo Saraiva e Barreto (2018), dentre todos esses bancos de dados, o mais utilizado é o MySQL (em seguida o PostgreSQL). E para esse gerenciador o PHP dispõe de duas bibliotecas para acesso ao banco de dados. A primeira e mais antiga, conhecida simplesmente como mysql, provê o suporte a versões do MySQL inferiores à versão 4.1.3, porém funciona também com as versões mais recentes, mas sem todas as funcionalidades disponíveis nessas versões.

O outro módulo para acesso ao MySQL, nas versões superiores a 4.1.3, é conhecido como Improved MySQL Extension, ou simplesmente mysqli. Essa versão, totalmente orientada a objetos, aproveita todas as funcionalidades novas existentes nas versões mais recentes do MySQL, tornando o acesso aos bancos de dados mais simples, rápido e seguro.

1.1.1 Obtendo o MySQL

Conseguir o MySQL é muito simples. Basta acessar o site oficial e baixar a versão MySQL Community Server, que é uma versão de download gratuito e o mais popular banco de dados *open source* do mundo.

Ele está disponível sob a licença GPL (*General Public License*) e recebe suporte de uma imensa e ativa comunidade de desenvolvedores *open source*.

É preciso escolher a plataforma na qual o MySQL vai rodar (temos Linux, Windows, Solaris e várias outras opções), baixar o instalador do MySQL e, já é possível, instalá-lo e utilizá-lo.

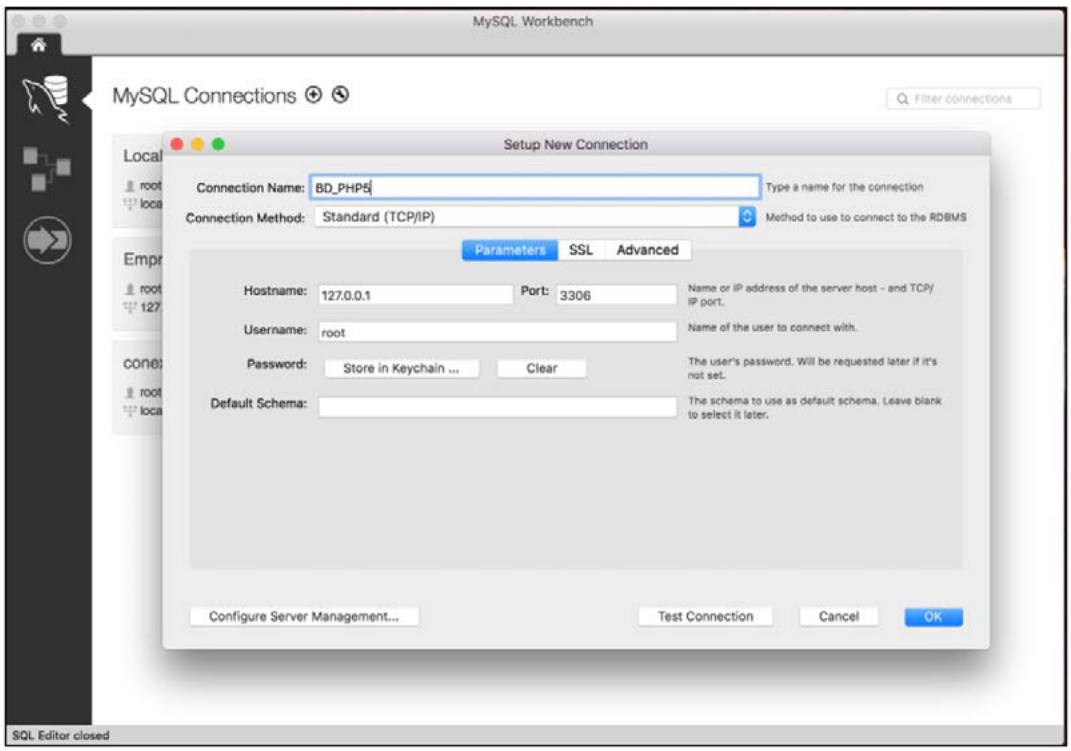
Para realizar a instalação, siga as instruções da documentação do próprio MySQL. No Windows, basta utilizar o instalador, disponível na página de downloads, e seguir as instruções dele. Já no Linux existem as opções de instalar os binários, por exemplo, em pacotes RPM (*Red Hat Package Manager*) e também a opção de baixar as fontes e compilá-los na máquina de destino do MySQL. **Classe**.

De acordo com Saraia e Barreto (2018), uma vez instalado o MySQL, para facilitar a sua utilização e desenvolvimento de código SQL, há uma ferramenta integrada de desenvolvimento chamada MySQL Workbench. O MySQL Workbench fornece recursos de modelagem e design de banco de dados, desenvolvimento SQL, administração de banco de dados e migração de banco de dados. O MySQL Workbench, assim como o MySQL Community Server, é um software sob licença GPL e *open source*. Para a utilização do PHP com MySQL o aplicativo EasyPHP deve ser iniciado, pois ele é o servidor PHP utilizado nesta disciplina.

1.1.2 Criando a base de dados

Para criar a base de dados no MySQL Workbench, inicie uma nova conexão no MySQL Connections, adicionando um nome para a conexão. Por padrão, conforme a Figura 1, o MySQL Workbench utiliza o *Hostname* 127.0.0.1, *Port* 3306, *Username* root e *Password* em branco.

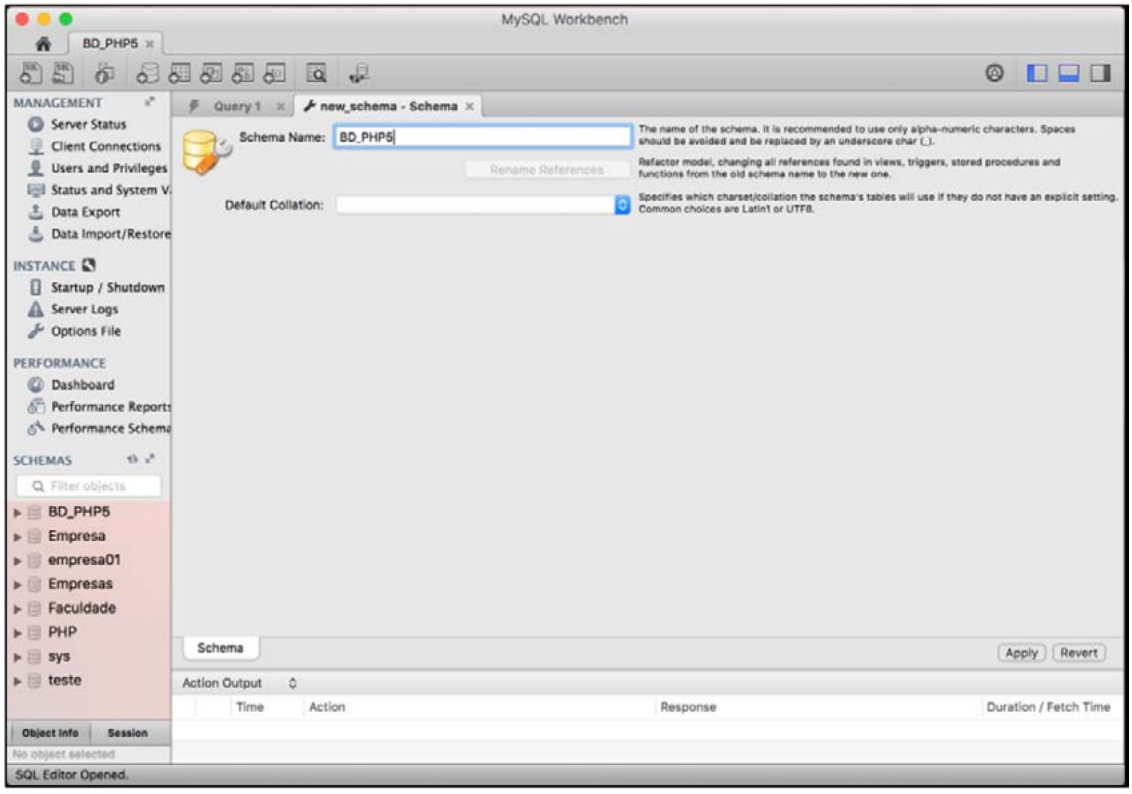
Figura 1 – Criação de conexão no MySQL



Fonte: o autor.

Após a criação da conexão, é necessário criar o banco de dados. Para isso, clique em “Criar um novo esquema” e atribua um nome ao banco de dados, como realizado na Figura 2, com nome BD_PHP5.

Figura 2 – Criação do banco de dados



Fonte: o autor.

Com a criação do banco de dados, crie a tabela usuário que será utilizada para os testes de conexão. A Figura 3 apresenta a criação da tabela Usuário, através da linguagem SQL, que possui os seguintes campos:

- userId: identificador do usuário, tipo inteiro, não nulo, valor autoincrementado e chave primária da tabela.
- userName: nome do usuário do tipo varchar com 30 caracteres.
- userLogin: login do usuário, tipo varchar com 20 caracteres e não nulo.
- userPassw: senha do usuário, tipo varchar com 20 caracteres e não nulo.
- userNivel: nível do usuário, tipo inteiro, não nulo e com valor padrão 1.
- userEmail: email do usuário e tipo varchar com 120 caracteres.

Figura 3 – Criação da tabela Usuário

```
CREATE TABLE Usuario(
userId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
userName VARCHAR(30),
userLogin VARCHAR(20) NOT NULL,
userPassw VARCHAR(20) NOT NULL,
userNivel INT NOT NULL DEFAULT 1,
userEmail VARCHAR (120)

);

unimed261874
```

Fonte: o autor.

1.1.3 Classe mysqli

A classe principal, contendo as funções básicas, tais como conexão com o servidor MySQL e seleção do banco de dados, gerenciamento de transações, busca de informações do banco, execução de consultas e encerramento da conexão (SOARES, 2013).

1.1.4 Classe mysqli_stmt

De acordo com Saraiva e Barreto (2018), essa classe disponibiliza um preparador de consultas para o MySQL, permitindo que sejam passados parâmetros para a consulta, ou seja, podemos preparar consultas padrão e quando for necessário, trocar os parâmetros de execução.

1.1.5 Classe mysqli_result

A classe mysqli_result deve ser utilizada para recuperação de consultas que retornem resultado (SELECT, SHOW, EXPLAIN e DESCRIBE). Nessa classe estão disponíveis os métodos para navegação no resultado e recuperação de registros.

1.1.6 Instância da Classe mysqli

A primeira ação executada é a conexão do servidor MySQL e a seleção do banco de dados. No mysqli isso é realizado em uma única etapa, informando o servidor, usuário, senha e banco de dados.

Qualquer erro gerado na conexão com o banco de dados pode ser recuperado pelas funções mysqli_connect_errno() e mysqli_connect_error(). Desta forma podemos testar o resultado da conexão de duas maneiras. A primeira verificando se a conexão não foi criada (algo como if(!$_con)) e a outra verificando se existe um erro após a conexão (if(mysqli_connect_errno!=0)).

A Figura 4 ilustra a variável $_con que recebe o retorno da classe mysqli, que se conecta ao banco de dados **banco**, no servidor localhost, usuário root e senha vazia. Se não for possível realizar a conexão, as classes mysqli_connect_errno() e mysql_connect_error() irão apresentar o erro de conexão.

Figura 4 – Conexão com o MySQL

```
<?php
$con = new mysqli("localhost", "root", "", "banco");
if(!$con) {
echo "Não foi possível conectar ao MySQL. Erro #" . mysqli_connect_errno() . "
: " . mysqli_connect_error();
exit;
}
?>
```

Fonte: o autor.

1.1.7 Query

Uma vez estabelecida a conexão com o banco de dados (ou seja, a classe foi instanciada com sucesso), de acordo com Saraiva e Barreto (2018),

podem ser executados os comandos SQL necessários. Para isso utiliza-se o método query da classe mysqli.

O número de registros afetados por um comando SQL (INSERT, UPDATE, DELETE) está armazenado no atributo affected_rows da classe mysqli. Para consultas que retornam um resultado (SELECT, SHOW, EXPLAIN e DESCRIBE), o método query() da classe mysqli retorna um objeto do tipo mysqli_result (uma instância dessa classe) e todas as propriedades e métodos referentes ao resultado estarão presentes nesse objeto. Há, por exemplo, o número de linhas retornado no atributo num_rows desse objeto, e para retornar uma linha da consulta, deve-se utilizar um dos métodos fetch disponíveis no objeto (fetch_array, fetch_assoc, fetch_row ou fetch_object). Qualquer erro retornado por uma consulta ao banco de dados estará definido nos atributos error e erro da classe mysqli.

Figura 5 – Exemplo de manipulação do banco de dados

```
<?php
$_con = new mysqli(“localhost”,”root”,”root”,”BD_PHP5”);
if(!$_con) {
// ou if(mysqli_connect_errno()!=0)
        echo “Não foi possível conectar ao MySQL. Erro #” . mysqli_
connect_errno() . “ : “ . mysql_connect_error(); exit;
}
// incluir alguns registros na tabela Usuario
$_sql = “INSERT INTO Usuario VALUES”;
$_sql .= “(NULL,’Darci F.Soares’,’Darci’,’teste’, 1, ‘darci@walace.com.br’),”;
$_sql .= “(NULL,’Elza M.S.Soares’,’Elza’,’teste’,2, ‘elza@walace.com.br’)”;
$_res = $_con->query($_sql);
if($_res===FALSE) {
echo “Erro na inclusão dos registros... “ . $_con->error . “<br/>”;
}
```

```
else {
echo $_con->affected_rows . " Registros incluídos com Sucesso<br/>";
}
// Agora vamos alterar alguns registros
$_sql = "UPDATE Usuario SET userNivel=2 WHERE userNivel=1";
$_res = $_con->query($_sql);
if($_res===FALSE) {
echo "Erro na alteração dos registros... " . $_con->error . "<br/>";
} else {
echo $_con->affected_rows . " Registros alterados<br/>";
}
// finalmente vamos listar os registros existentes no banco
$_sql = "SELECT * FROM Usuario";
$_res = $_con->query($_sql);
 if($_res===FALSE) {
echo "Erro na consulta... " . $_con->error . "<br/>";
}
else {
$_nr = $_res->num_rows;
echo "A consulta retornou " . (int) $_nr . " registro(s)<br/>";
if($_nr>0) {
// Primeiro o cabeçalho com os campos da tabela
echo "<table border=1>";
echo "<tr bgcolor='#f0f0f0'>";
for($_i=0;$_i<$_res->field_count;$_i++) {
    $_f = $_res->fetch_field_direct($_i);
    echo "<td>" . $_f->name . "</td>";
}
```

```
echo “</tr>”;
// Agora o resultado
while($_row=$_res->fetch_assoc()) {
  echo “<tr>”;
  foreach($_row as $_vlr) {
    echo “<td>$_vlr</td>”; echo “</tr>”;
  }
  echo “</table>”;
 }
}
$_con->close();
?>
```

Fonte: o autor.

Dessa forma, você aprendeu a realizar a instalação do MySQL. Você também aprendeu a criar banco de dados no MySQL e a utilizar o PHP com MySQL.

TEORIA EM PRÁTICA

Você é um programador PHP e está criando uma página de contato em um website de uma imobiliária. Já fez toda a estrutura e criação da página de contato em HTML e agora gostaria de realizar o armazenamento em banco de dados. Porém, ficou na dúvida de como realizar o armazenamento dos dados de contato em um banco de dados. Pesquise sobre como realizar o armazenamento dos dados de contato em um banco de dados.

VERIFICAÇÃO DE LEITURA

1. A classe principal, contendo as funções básicas, tais como conexão com o servidor MySQL e seleção do banco de dados, gerenciamento de transações, busca de informações do banco, execução de consultas e encerramento da conexão. Assinale a alternativa que apresenta, corretamente, essa classe principal:

   a. mysqli

   b. mysqli_stmt

   c. mysqli_result

   d. mysqli_connect

   e. mysqli_conn

2. Uma vez estabelecida a conexão com o banco de dados (ou seja, a classe foi instanciada com sucesso), podem ser executados os comandos SQL necessários. Assinale a alternativa que apresenta, corretamente, o método utilizado:

   a. Connect

   b. Query

   c. mysqli_result

   d. mysqli_stmt

   e. Conn

3. Uma vez finalizada as transações SQL, a conexão necessita ser desconectada do banco de dados. Assinale a alternativa que apresente, corretamente, o método utilizado para desconectar:

   a. disconnect()

   b. close()

   c. stop()

   d. exit()

   e. done()

Referências bibliográficas

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

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.

Gabarito

Questão 1 – Resposta: A

Resolução: A classe mysqli apresenta funções básicas, tais como, conexão com o servidor MySQL e seleção do banco de dados, gerenciamento de transações, busca de informações do banco, execução de consultas e encerramento da conexão.

Feedback de reforço: Classe mysql com as funções básicas.

Questão 2 – Resposta: B

Resolução: Uma vez estabelecida a conexão com o banco de dados (ou seja, a classe foi instanciada com sucesso), podem ser executados os comandos SQL necessários através do método Query.

Feedback de reforço: Método para utilização de sentenças.

Questão 3 – Resposta: B

Resolução: Uma vez finalizada as transações SQL, a conexão necessita ser desconectada do banco de dados e este processo é realizado através do método close().

Feedback de reforço: Método para fechar conexão com banco de dados.

Bons estudos!