Introdução a linguagem PHP

Apresentação

O *Hypertext Preprocessor* (PHP) é uma linguagem de programação que opera no lado servidor (*server-side*), diferentemente de HTML, CSS e JavaScript, que operam no lado do cliente, no navegador (*client-side*). O PHP possibilita o desenvolvimento de sistemas *web* completos e dinâmicos, oferecendo ao programador um amplo conjunto de recursos.

Nesta Unidade de Aprendizagem, você vai ver as possibilidades oferecidas pela linguagem e também as suas potencialidades, a partir dos diferentes recursos.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Definir os aspectos básicos da linguagem de programação PHP.
- Empregar os diferentes recursos oferecidos pela linguagem PHP.
- Identificar como são produzidos os sistemas *web* baseados em um modelo cliente-servidor.

Infográfico

Com a linguagem PHP, é possível criar aplicações *web*, em que *scripts* são geralmente interpretados por um servidor *web*. Para isso, o servidor precisa ter um interpretador PHP corretamente configurado.

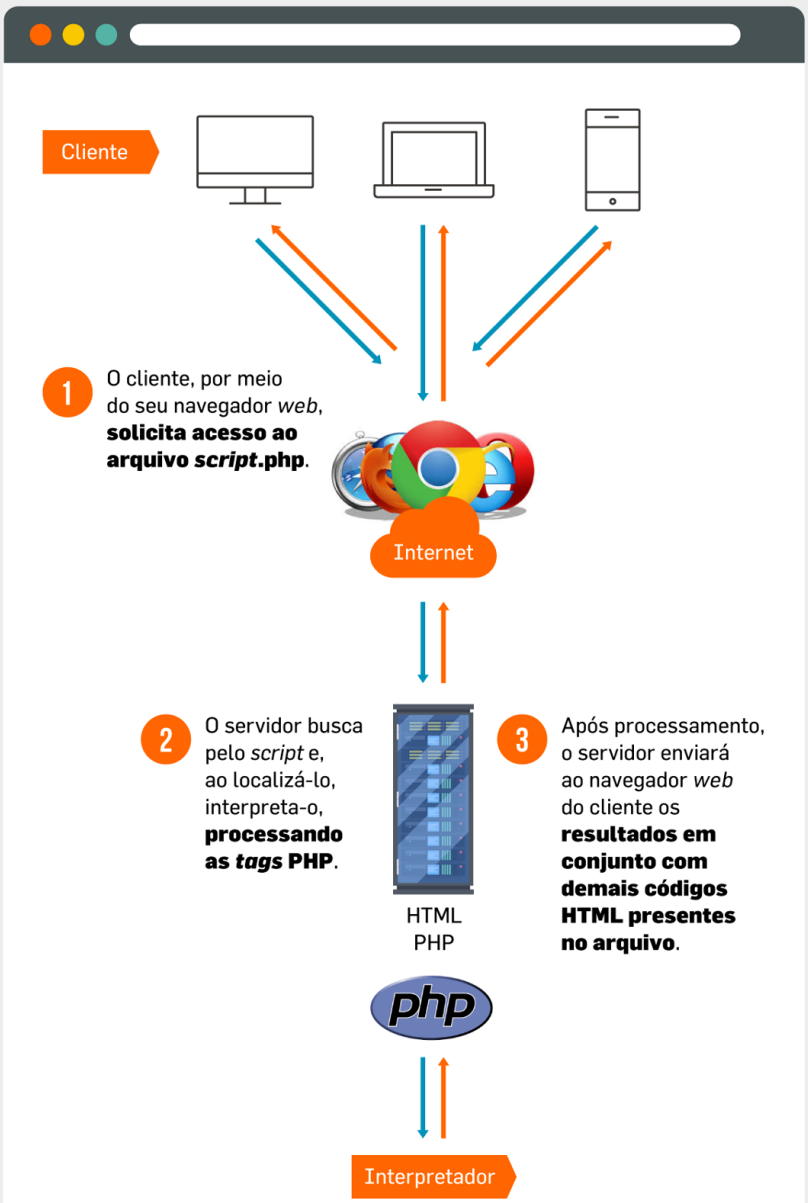
Confira, no Infográfico a seguir, como a linguagem PHP é pré-processada no servidor para que os resultados das aplicações *web* sejam exibidos no navegador dos clientes.

ESTRUTURA NECESSÁRIA PARA O FUNCIONAMENTO DO PHP

Existe toda uma estrutura e um processo para que as informações de uma página PHP sejam exibidas para o cliente.

O que acontece quando um cliente (navegador) solicita uma página PHP para o servidor? Qual sequência essa solicitação percorre para retornar a página de resposta ao cliente?

Confira a resposta para essas perguntas a seguir.



Como ilustrado, o processamento das informações acontece no lado do servidor, enquanto o cliente fica responsável por enviar as solicitações e receber as respostas de volta. A partir desse estudo, é possível verificar que os navegadores *web* dos clientes não têm conhecimento sobre os códigos previamente existentes em PHP, pois as páginas somente serão recebidas após serem devidamente processadas pelo servidor.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Conteúdo do Livro

PHP é uma das linguagens de programação mais populares da web, usada para criar sites e aplicativos dinâmicos. É uma linguagem interpretada de código aberto, o que significa que é gratuita para uso e fácil de aprender. A sintaxe do PHP é semelhante a outras linguagens de programação, como C e Java, mas é especialmente projetada para a criação de sites e aplicativos da web. PHP é capaz de se comunicar com bancos de dados, gerar conteúdo dinâmico e interagir com outros serviços da web, tornando-a uma ferramenta poderosa para o desenvolvimento de aplicações web.

No capítulo Introdução à linguagem PHP, base teórica desta Unidade de Aprendizagem, você vai estudar os conceitos básicos de programação em PHP e seus recursos. Além disso, vamos entender como os sistemas web baseados em um modelo cliente-servidor funcionam.

FERRAMENTAS DE DESENVOLVIMENTO *WEB*

sagah

Introdução à linguagem PHP

Rafael Albuquerque Pinto

OBJETIVOS DE APRENDIZAGEM

- Definir os aspectos básicos da linguagem de programaçãa PHP.
- Empregar os diferentes recursos oferecidos pela linguagem PHP.
- Identificar como são produzidos os sistemas *web* baseados em um modelo cliente-servidor.

Introdução

A linguagem PHP é uma das mais populares na criação de páginas *web* dinâmicas. É uma linguagem de programação do lado do servidor, o que significa que as páginas são processadas no servidor antes de serem enviadas para o navegador do usuário. Essa tecnologia permite a criação de *sites* dinâmicos e interativos, que respondem aos usuários em tempo real.

Para começar a aprender PHP, é necessário ter um entendimento básico de programação. A linguagem utiliza conceitos como variáveis, *loops* e funções, comuns em outras linguagens de programação. Sua sintaxe é simples e objetiva, o que a torna acessível para desenvolvedores iniciantes. Além disso, é uma linguagem de código aberto, o que significa que é possível utilizá-la sem custos, bem como contribuir com sua melhoria e desenvolvimento.

Neste capítulo, você vai estudar a linguagem de programaçãa PHP, seus principais conceitos e aspectos, além dos seus diferentes recursos. Vai, também, acompanhar como são produzidos os sistemas *web* baseados em um modelo cliente-servidor.

Conceitos e aspectos básicos da PHP

PHP é uma linguagem de programação de código aberto muito popular e amplamente utilizada para o desenvolvimento de aplicações *web* dinâmicas. Foi criada por Rasmus Lerdorf em 1994 e era inicialmente chamada de Personal Home Page Tools (Ferramentas para Página Pessoal). Mais tarde, com o lançamento de sua versão 3.0 em 1997, passou a ser conhecida como PHP: Hypertext Preprocessor (Pré-processador de Hipertexto PHP) (ACHOUR *et al.*, 2022; LERDORF; TATROE; MACINTYRE, 2006).

Desde então, a linguagem PHP passou por várias atualizações e melhorias, incluindo a introdução do paradigma de programação orientada a objetos (POO) na versão PHP 5. Confira no Quadro 1 algumas das versões mais importantes desde sua criação.

Quadro 1. Versões da PHP

Versão	Descrição	Data de lançamento
PHP 3	Primeira versão independente da PHP.	junho de 1998
PHP 4	Introduziu muitos recursos, incluindo suporte a objetos.	maio de 2000
PHP 5	Incluiu melhorias significativas na orientação a objetos e suporte para novos recursos, como *namespaces*.	julho de 2004
PHP 7	Introduziu melhorias significativas no desempenho e na segurança da linguagem.	dezembro de 2015
PHP 8	Lançou muitas funcionalidades, incluindo suporte a tipos de retorno e argumentos mais precisos, além de melhorias no desempenho e na segurança.	novembro de 2020

Fonte: Adaptado de Achour *et al.* (2022).

A PHP é uma linguagem de programação conhecida por suas características únicas e úteis, como sua sintaxe simples e familiar, que facilita a aprendizagem daqueles que têm conhecimento básico de programação. Além disso, a sintaxe da PHP é semelhante a outras linguagens populares, como Java e C, tornando a transição de uma linguagem para outra mais fácil. Outra característica

importante da PHP é seu suporte para diferentes bancos de dados populares, como MySQL, PostgreSQL e SQLite, permitindo que os desenvolvedores criem aplicativos *web* com recursos avançados de gerenciamento de dados (ACHOUR *et al*., 2022; WELLING; THOMSON, 2016).

Para acelerar o processo de desenvolvimento, há muitas bibliotecas e *frameworks* disponíveis para a PHP, como o Laravel, um *framework* PHP popular que oferece recursos avançados para o desenvolvimento de aplicações *web*, simplificando muitas tarefas comuns de programação. Além disso, a PHP é conhecida por sua escalabilidade e seu desempenho, podendo ser escalada para atender às necessidades de grandes aplicativos *web*. É rápida e eficiente, o que a torna uma escolha popular para o desenvolvimento de aplicações *web* de alto desempenho. Por fim, é uma linguagem *open source*, o que significa que é gratuita e pode ser modificada e aprimorada pela comunidade de desenvolvedores. Isso levou ao desenvolvimento de muitos recursos e bibliotecas de terceiros para a PHP, aumentando sua funcionalidade (KHERLAKIAN, 2020; LARAVEL, 2023).

Modelo de aplicação *web* cliente-servidor

Um modelo de aplicação *web* cliente-servidor é um tipo de arquitetura de *software* que envolve a comunicação entre um cliente e um servidor para fornecer serviços pela internet. Esse modelo tem sido amplamente utilizado no desenvolvimento de aplicações *web*, e a PHP é uma das linguagens de programação mais populares para a construção de aplicações *web* baseadas em cliente-servidor (TANENBAUM; WETHERALL, 2011).

Esse modelo de aplicação *web* é composto por dois componentes principais: o cliente e o servidor. O cliente é a interface com o usuário e é responsável por enviar solicitações ao servidor, e o servidor processa as solicitações do cliente e envia uma resposta de volta — normalmente a PHP é responsável por essa tarefa. O cliente pode ser qualquer dispositivo que tenha um navegador *web*, como um computador, *tablet* ou *smartphone*. Já o servidor é responsável por armazenar e processar os dados e a lógica de negócios necessários para atender às solicitações do cliente. O servidor pode ser executado em um computador remoto ou em um servidor na nuvem (TANENBAUM; WETHERALL, 2011).

O modelo de aplicação *web* cliente-servidor é altamente escalável, permitindo que várias solicitações de clientes sejam processadas simultaneamente pelo servidor. Ele também é flexível, o que faz com que diferentes tipos de clientes (por exemplo, navegadores *web* em dispositivos móveis) possam se conectar ao mesmo servidor e solicitar serviços.

Entre as características da PHP que a tornam uma escolha popular para o desenvolvimento de aplicações *web* baseadas em modelo de aplicação *web* cliente-servidor, estão a facilidade de aprendizado e uso, a compatibilidade com diferentes plataformas, o suporte a bancos de dados e uma grande comunidade de desenvolvedores que oferecem suporte e recursos para ajudar no desenvolvimento de aplicações *web*.

Geralmente, as aplicações *web* desenvolvidas em PHP seguem o modelo de arquitetura cliente-servidor. Nesses casos, o servidor é responsável pelo processamento de solicitações e envio de respostas para o cliente. As solicitações dos clientes podem incluir informações de formulários, URLs ou dados de *cookies*, que são enviados ao servidor para processamento. O servidor, por sua vez, executa *scripts* PHP, que manipulam esses dados e geram uma resposta em HTML ou em outro formato compatível com o navegador do cliente.

Fique atento

A PHP é frequentemente usada em combinação com outras tecnologias *web*, como HTML, CSS e JavaScript, para criar aplicações *web* dinâmicas e interativas. Os desenvolvedores podem usar bibliotecas e *frameworks* em PHP, como Laravel, CodeIgniter e Yii, para simplificar o processo de desenvolvimento de aplicações *web*.

Recursos oferecidos pela linguagem PHP

Como vimos, a linguagem PHP oferece diversos recursos para o desenvolvimento de aplicações *web*. Nesta seção, vamos ver com mais detalhes alguns desses recursos, como a possibilidade de criar páginas dinâmicas, capazes de processar informações e gerar conteúdo de acordo com as solicitações do usuário; o suporte a diversos bancos de dados, como MySQL, PostgreSQL e Oracle; a possibilidade de integração com outras tecnologias, como HTML, CSS e JavaScript; bibliotecas de funções, que facilitam o desenvolvimento de aplicações *web*; a facilidade de aprendizado e utilização.

Para iniciar um bloco de código PHP, utiliza-se o delimitador `<?php` e, para finalizar, utiliza-se o delimitador `?>`. Um exemplo de código PHP básico é o seguinte:

```
<?php

    echo "Olá, mundo!";

?>
```

A linguagem PHP é capaz de manipular diversos tipos de dados, permitindo que os desenvolvedores criem aplicações *web* dinâmicas e interativas, incluindo:

- números inteiros (exemplo: 10);
- números decimais (exemplo: 3,14);
- *strings* (exemplo: `"hello world"`);
- booleanos (*true* ou *false*);
- *arrays* (listas de valores);
- objetos (instâncias de classes);
- recursos (exemplo: conexões com bancos de dados);
- `null` (ausência de valor).

Os números inteiros e decimais são utilizados para representar valores numéricos, enquanto as *strings* são usadas para armazenar texto e caracteres. Os booleanos são usados para representar valores lógicos verdadeiros ou falsos. Os *arrays* permitem armazenar uma lista de valores em uma única variável, enquanto os objetos são instâncias de classes que têm propriedades e métodos. Já os recursos são usados para representar conexões com bancos de dados e outros recursos externos. Por fim, o valor `null` é utilizado para representar a ausência de valor em uma variável. Todos esses tipos de dados suportados pela linguagem PHP são essenciais para a criação de aplicações *web* complexas e eficientes.

Confira a seguir os comandos da linguagem PHP para definição de variáveis, constantes, operadores e estruturas de controle.

Variáveis

Para definir uma variável, utilizamos o símbolo `$`, seguido do nome da variável. Por exemplo:

```
<?php

                              $nome = "João";

$idade = 30;

?>
```

Constantes

Para definir uma constante, utilizamos a função `define()`, seguida do nome da constante e do seu valor. Por exemplo:

```
<?php

                    define("PI", 3.14);

?>
```

Operadores

Operadores são responsáveis pela realização de operações matemáticas, lógicas e de comparação. Na linguagem de programação, os principais operadores são os aritméticos e os de comparação.

- Operadores aritméticos: utilizados para realizar operações matemáticas em PHP, como adição (+), subtração (–), multiplicação (*), divisão (/) e módulo (%). Por exemplo:

```
<?php

      $a = 5;

      $b = 2;


      $soma = $a + $b; // Resultado: 7

      $subtracao = $a - $b; // Resultado: 3

      $multiplicacao = $a * $b; // Resultado: 10

      $divisao = $a / $b; // Resultado: 2.5

      $modulo = $a % $b; // Resultado: 1

?>
```

- Operadores de comparação: utilizados para comparar valores em PHP, como igual (==), diferente (!=), maior que (>), menor que (<), maior ou igual que (>=) e menor ou igual que (<=). Por exemplo:

```
<?php
	$a = 5;
	$b = 2;

	if ($a > $b) {
		echo "A é maior que B";
	}
?>
```

Estruturas de controle

As estruturas de controle são utilizadas para controlar o fluxo de execução do código, permitindo que determinadas ações sejam executadas de acordo com condições específicas. Confira a seguir as principais estruturas de controle na linguagem de programação.

- `if`/`else`: permite que uma ação seja executada caso uma condição seja verdadeira e que outra ação seja executada caso a condição seja falsa. Por exemplo: verificar se uma idade é maior do que 18.

```
<?php
	if ($idade >= 18) {
		echo “Você é maior de idade”;
	} else {
		echo “Você é menor de idade”;
	}
?>
```

- `switch`: permite que diferentes ações sejam executadas de acordo com o valor de uma variável. Por exemplo: verificar o dia da semana.

```
<?php
switch ($diaSemana) {
    case 1:
        echo “Hoje é segunda-feira”;
        break;
    case 2:
        echo “Hoje é terça-feira”;
        break;
    case 3:
        echo “Hoje é quarta-feira”;
        break;
    case 4:
        echo “Hoje é quinta-feira”;
        break;
    case 5:
        echo “Hoje é sexta-feira”;
        break;
    case 6:
        echo “Hoje é sábado”;
        break;
    case 7:
        echo “Hoje é domingo”;
        break;
}
?>
```

- `for`: permite que um bloco de código seja executado determinadas vezes. Por exemplo: imprimir de 1 a 10.

```
<?php
for ($i = 1; $i <= 10; $i++) {
    echo $i . "<br>";
}
?>
```

- `foreach`: permite que um bloco de código seja executado para cada elemento de um *array*. Por exemplo: imprimir cada elemento de um *array*.

```
<?php
$frutas = array("maçã", "banana", "laranja", "uva");
foreach ($frutas as $fruta) {
    echo $fruta . "<br>";
}
?>
```

- `while`: permite que um bloco de código seja executado enquanto uma condição for verdadeira. Por exemplo: imprimir números de 1 a 5.

```
<?php
$i = 1;
while ($i <= 5) {
    echo $i . "<br>";
    $i++;
}
?>
```

- `do/while`: permite que um bloco de código seja executado pelo menos uma vez e enquanto uma condição for verdadeira. Por exemplo: pedir que o usuário insira um número pelo menos uma vez.

```
<?php

do {

    $num = readline("Insira um número: ");

} while ($num <= 0);

echo "O número inserido foi: " . $num;

?>
```

Em resumo, a PHP oferece uma série de recursos e facilidades para o desenvolvimento de aplicações *web* dinâmicas, incluindo sintaxe básica intuitiva, suporte a diversos tipos de dados, operadores e estruturas de controle de fluxo.

Sistemas *web* baseados em um modelo cliente-servidor

Para construir uma aplicação *web* em PHP, é necessário ter um servidor *web* instalado e configurado, como o Apache, o Nginx e o IIS. É necessário também ter um editor de texto para escrever o código PHP, como o Sublime Text, o Visual Studio Code e o Atom.

Um exemplo simples de aplicação *web* em PHP sem banco de dados é uma aplicação que permite ao usuário fazer *upload* de arquivos para o servidor. Nesse exemplo, a aplicação é construída usando o modelo cliente-servidor, em que, como vimos, o navegador do cliente solicita uma página para o servidor, que, por sua vez, responde com a página HTML ou conteúdo solicitado.

Para construir essa aplicação, é necessário criar uma estrutura básica de arquivos e pastas. Por exemplo, pode-se criar uma pasta chamada “meuapp” na raiz do servidor *web* e, dentro dela, criar uma pasta chamada “public”, para armazenar os arquivos públicos da aplicação (como imagens, arquivos CSS e JavaScript), e uma pasta chamada “uploads”, para armazenar os arquivos enviados pelos usuários. O próximo passo é criar um arquivo “index.php” na pasta “public”. Esse arquivo será a página inicial da aplicação e será exibido

quando o usuário acessar o URL da aplicação. No arquivo “index.php”, pode-se incluir o código HTML básico, bem como o código PHP necessário para lidar com o envio de arquivos. Como exemplo, o seguinte código pode ser usado para exibir um formulário simples que permite ao usuário enviar um arquivo para o servidor:

```
<!DOCTYPE html>

<html>

<head>

    <title>Minha Aplicação Web</title>

</head>

<body>

    <h1>Enviar arquivo</h1>

   <form action="upload.php" method="post" enctype="multipart/
form-data">

        Selecione um arquivo para enviar:

        <input type="file" name="arquivo">

        <br>

        <br>

        <input type="submit" value="Enviar">

    </form>

</body>

</html>
```

Esse código inclui um título de página, um cabeçalho e um formulário que permite ao usuário selecionar um arquivo para enviar ao servidor. Quando o usuário clica no botão “Enviar”, o formulário é enviado para o arquivo “upload.php” no servidor. Em seguida, é necessário criar o arquivo “upload.php” na pasta “public”. Esse arquivo será responsável por receber o arquivo enviado pelo usuário e salvá-lo na pasta “uploads”. O código PHP necessário para fazer isso é relativamente simples:

```
<?php

if(isset($ _ FILES['arquivo'])) {

    $arquivo = $ _ FILES['arquivo'];

    $nome = $arquivo['name'];

    $tmp _ nome = $arquivo['tmp _ name'];

    $error = $arquivo['error'];

    if($error === UPLOAD _ ERR _ OK) {

      move _ uploaded _ file($tmp _ nome, 'uploads/' . $nome);

        echo 'Arquivo enviado com sucesso!';

    } else {

        echo 'Ocorreu um erro ao enviar o arquivo. Tente
novamente mais tarde.';

    }

}

?>
```

Esse código verifica se um arquivo foi enviado pelo usuário e, se sim, move o arquivo temporário para a pasta "uploads" com o nome original do arquivo. Se ocorrer algum erro durante o envio do arquivo, a aplicação exibe uma mensagem de erro. Para executar a aplicação, é necessário iniciar o servidor *web* e acessar o URL da aplicação no navegador. Por exemplo, se o servidor *web* estiver rodando localmente na porta 8000, pode-se acessar a aplicação digitando "http://localhost:8000" na barra de endereços do navegador. Quando o usuário seleciona um arquivo para enviar, a aplicação envia o arquivo para o servidor e exibe uma mensagem informando se o arquivo foi enviado com sucesso ou se houve algum erro durante o envio. A Figura 1 mostra a tela da aplicação de selecionar um arquivo para enviar ao servidor.

Figura 1. Página "index.php" responsável por selecionar o arquivo.

O exemplo que estudamos nesta seção é de uma aplicação *web* em PHP sem banco de dados, que permite ao usuário enviar arquivos para o servidor. É importante notar que esse exemplo não inclui nenhuma medida de segurança ou validação de entrada, o que pode ser um problema de segurança. Além disso, para uma aplicação mais complexa, pode ser necessário usar um *framework* PHP ou biblioteca para lidar com tarefas comuns, como validação de formulários, autenticação de usuários e manipulação de arquivos. No entanto, esse exemplo serve como um ponto de partida para quem está começando a aprender PHP e deseja construir uma aplicação *web* simples. Com um pouco mais de estudo e prática, é possível construir aplicações mais complexas e úteis que atendam às necessidades dos usuários.

Neste capítulo, vimos que, desde seu lançamento em 1995, a linguagem PHP tem sido uma das principais escolhas para desenvolvedores que trabalham na *web*. Ela foi criada para ser fácil de usar e se integrar bem com HTML e outras tecnologias de *front-end*. Com o tempo, a linguagem evoluiu e, hoje, conta com muitos recursos avançados, como orientação a objetos e manipulação de bancos de dados.

Referências

ACHOUR, M. *et al*. Manual da PHP. *PHP*, 2022. Disponível em: https://www.php.net/manual/pt_BR/index.php. Acesso em: 22 mar. 2023.

KHERLAKIAN, M. What is PHP? *Zend*, 2020. Disponível em: https://www.zend.com/blog/what-is-php. Acesso em: 22 mar. 2023.

LARAVEL. The PHP framework for web artisans. *Laravel*, 2023. Disponível em: https://laravel.com/. Acesso em: 22 mar. 2023.

LERDORF, R.; TATROE, K.; MACINTYRE, P. A brief history of PHP. *In*: LERDORF, R.; TATROE, K.; MACINTYRE, P. (ed.). *Programming PHP*. 2. ed. Sebastopol: O'Reilly, 2006. cap. 1.

TANENBAUM, A. S.; WETHERALL, D. J. *Computer networks*. 5. ed. Boston: Pearson Education, 2011.

WELLING, L.; THOMSON, L. *PHP and MySQL web development*. 5. ed. Boston: Addison-Wesley Professional, 2016.

Leituras recomendadas

DALL'OGLIO, P. *PHP*: programando com orientação a objetos. 2. ed. São Paulo: Novatec, 2009.

LOCKHART, J. *PHP moderno*: novos recursos e boas práticas. São Paulo: Novatec, 2015.

NIEDERAUER, J. *Desenvolvendo websites com PHP*. 3. ed. São Paulo: Novatec, 2016.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Conteúdo:

sagah

Dica do Professor

A linguagem PHP opera no lado do servidor, permitindo o desenvolvimento de aplicações *web* completas e dinâmicas. Ela oferece ao desenvolvedor um grande conjunto de recursos, como visualização de dados, estruturas condicionais e de repetição e estruturas de dados, como o *array*, etc.

Nesta Dica do Professor, confira como o PHP se comporta na transição dos dados entre servidor e navegador, além das possibilidades oferecidas por essa linguagem.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **O PHP permite grande flexibilidade a rapidez no desenvolvimento de aplicações *web*.**

O que é correto afirmar sobre o PHP?

A) É uma linguagem de *script* que roda e faz o processamento do lado do cliente para rodar.

B) É uma linguagem lógica específica para cliente/servidor.

C) É uma linguagem de *script* que roda do lado do servidor e faz o processamento no lado do cliente.

D) É uma linguagem de *script* que executa um executável no lado do servidor.

E) É uma linguagem de *script* que roda e faz todo o processamento do lado do servidor.

2) **Na linguagem PHP, é possível a criação de variáveis de diversos tipos de dados.**

Como se declara uma variável em PHP?

A) Usa-se o sinal de $, e damos dar um espaço entre o nome da variável.

B) Deve-se colocar o tipo de dados antes da declaração da variável.

C) Usa-se o sinal de $ antes do nome da variável.

D) Coloca-se o tipo de dados antes e, em seguida, o sinal de $ mais o nome da variável.

E) Define-se uma variável da forma que desejar, e o PHP entende.

3) **A linguagem PHP trabalha com alguns tipos de dados, entre eles inteiro, ponto flutuante, *string* e *array*.**

Avalie as asserções a seguir e a relação proposta entre elas:

I. É obrigatório colocar o tipo de dados na declaração de uma variável em PHP.

PORQUE

II. O interpretador PHP resolve o tipo de dados dinamicamente.

A respeito dessas asserções, assinale a opção correta.

A) A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.

B) As asserções I e II são proposições verdadeiras, e a II é uma justificativa da I.

C) A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.

D) As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa da I.

E) As asserções I e II são ambas proposições falsas.

4) Os operadores de atribuição são utilizados quando se deseja atribuir valores em expressões ou em comandos decisão, repetição, entre outros.

Quais são os operadores de atribuição corretos em PHP?

A) =, <=, -=, !=.

B) =, +=, /=, .=.

C) <>, >=, *=, <.

D) &&, ||, -=, =.

E) -=, +=, %, !=.

5) O PHP trabalha com comandos de *loop*. Um desses comandos de *loop* é o *for*.

O que faz o comando de *loop for*?

A) Tem a função de repetir determinado código em um número indeterminado.

B) Tem a função de repetir determinado código em um número de vezes conhecido.

C) Tem a função de realizar testes condicionais de expressões lógicas.

D) Tem a função de executar determinado código em um número de vezes desconhecido.

E) Tem a função de saber se determinado código tem um número de vezes conhecido.

Na prática

A linguagem PHP trabalha com comandos de *loop*, isto é, comandos que executam determinado trecho de código a quantidade de vezes que o usuário necessitar ou até que determinado critério de parada seja satisfeito. Um desses comandos é o *for*.

Acompanhe, Na Prática, a história de Paulo, estudante de computação, que desenvolveu um sistema em PHP para calcular a tabuada usando o comando *for*.

CRIANDO UMA APLICAÇÃO SIMPLES DE TABUADA EM PHP

Paulo é desenvolvedor PHP iniciante. Para treinar seus conhecimentos, está tentando criar uma aplicação simples nessa linguagem para calcular a tabuada.

Para isso, ele usará o ***for***, um dos comandos de *loop* mais utilizados em PHP.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Comando *for*

Esse comando utiliza um contador interno e executa a repetição de um bloco de código em um número conhecido de vezes.

Confira, a seguir, como Paulo executou essa tarefa.

Sintaxe do comando *for*

A sintaxe do comendo for é a seguinte:

```
for (inicialização; condição; operador){
   // trecho de código a ser repetido
}
```

Para calcular a tabuada de um número, a aplicação de Paulo precisará **executar 11 vezes o cálculo da multiplicação**, ou seja:

```
for ($i=0; $i<11; $i++){
   // cálculo da multiplicação
}
```

Código php desenvolvido

O código desenvolvido fica embutido em um documento HTML, entre as *tags* **<?php** e **?>**.

O programa faz a leitura de um número inteiro. Em seguida, esse número é multiplicado por 0, e o resultado é impresso na tela. Depois, é multiplicado por 1, e novamente o resultado é exibido; depois, é multiplicado por 2, e assim sucessivamente.

calculo_tabuada.php

```
<!DOCTYPE html>
<html>
<body>
    <form name="form" method="get">
        Informe um número de 1 a 10: <input type="text" value="" name="numero">
        <input type = "submit" value = "Calcular">
    </form>
<?php
    $numero = (int)$_GET [numero];
    for ($i = 0; $i < 11; $i++)
    {
        echo " $i x $numero = " . $i*$numero . "<br>";
    }
?>
</body>
</html>
```

O comando *for* tem sua inicialização com a variável *i* valendo zero; em seguida, essa variável é testada até determinada condição, no caso *i* < 11. Se o teste é verdadeiro, e o resultado da tabuada é impresso na tela, incrementa-se a variável *i* em um, e a condição é testada novamente. Quando a condição se tornar falsa, o *loop* é encerrado.

Resultado

Se um número for digitado na caixa de texto e o botão **Calcular** for pressionado, será mostrada a tabuada de 0 a 10 desse número.

Veja como fica a execução para o número 7 dessa aplicação desenvolvida por Paulo:

Informe um número: 7 Iniciar

0 x 7 = 0
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
5 x 7 = 35
6 x 7 = 42
7 x 7 = 49
8 x 7 = 56
9 x 7 = 63
10 x 7 = 70

Conclusão

Portanto, é possível verificar que Paulo concluiu com êxito seu objetivo.
Ele trabalhou com um comando de *loop*, que executou determinado trecho de código até que uma condição de parada fosse satisfeita.

Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

PHP 8 (Guia de atualização da linguagem)

Neste vídeo, você pode verificar as principais características da versão 8 da linguagem PHP.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Manual do PHP

Neste *link*, confira a documentação oficial sobre a linguagem PHP, da introdução à instalação e à configuração.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Tutorial PHP

Neste *link*, você confere um tutorial do PHP e pode aprender de forma simples e fácil com o PHP Tryit.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.