



Programação de páginas dinâmicas com PHP

Você será introduzido à programação dinâmica de páginas HTML com o uso da linguagem de programação PHP. Vamos abordar, também, os principais conceitos de programação, uso de variáveis, estruturas de decisão, estruturas de repetição, arrays e funções em PHP.

Prof. Alexandre de Oliveira Paixão

1. Itens iniciais

Preparação

Para melhor compreensão do conteúdo, é recomendado um conhecimento básico de lógica de programação e de HTML.

Para a execução dos exemplos em PHP de maneira externa a este material e para os exemplos contendo tags HTML, será necessário um editor de texto com suporte à marcação PHP e um servidor web com suporte à linguagem. Em relação ao editor, no sistema operacional Windows é indicado o Notepad++. No Linux, o Nano Editor. Quanto ao servidor, recomenda-se o Apache.

A fim de utilizar os códigos-fontes originais propostos neste estudo para o aprendizado de PHP, baixe [aqui](#) o arquivo com os códigos e descompacte-o em seu dispositivo.

Objetivos

- Examinar a linguagem de programação PHP e seus conceitos básicos.
- Aplicar as estruturas de decisão e de repetição disponíveis em PHP.
- Identificar conceitos relativos a vetores e funções em PHP.

Introdução

Neste vídeo, você vai conhecer o conceito de programação dinâmica em páginas HTML e entenderá como podemos usar a linguagem de programação PHP para atingir esse objetivo.



Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A linguagem PHP

PHP é uma linguagem baseada em scripts, que possibilita o desenvolvimento de aplicações do lado do servidor, gerando páginas web interativas e dinâmicas, ou seja, com conteúdos personalizados. Além disso, PHP é gratuita e bastante difundida, de modo que é possível encontrar muitas referências a respeito. PHP ainda possibilita o uso do paradigma de orientação a objetos, é fracamente tipada, permite a integração com diferentes bancos de dados, pode ser usada para controle de acesso, entre outros recursos. Outra característica importante é a possibilidade de se utilizarem scripts PHP diretamente em páginas HTML.

Neste vídeo, exploramos como PHP, uma linguagem de scripts gratuita e amplamente difundida, possibilita o desenvolvimento de aplicações do lado do servidor e gera páginas web interativas. Descubra seus recursos, como orientação a objetos, integração com diferentes bancos de dados, controle de acesso e facilidade de inserir scripts PHP diretamente em HTML. Ideal para quem quer criar conteúdos web personalizados e dinâmicos.



Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O **PHP** é uma linguagem de script open source de uso geral. Muito utilizada, é especialmente adequada para o **desenvolvimento Web** e pode ser embutida dentro do **HTML (PHP)**.

PHP

Hypertext Preprocessor.

A explicação anterior consta no site oficial do PHP, de onde os fragmentos a seguir também foram retirados:



O que distingue o PHP de algo como o JavaScript no lado do cliente é que o código é executado no servidor, gerando o HTML que é então enviado para o navegador. O navegador recebe os resultados da execução desse script, mas não sabe qual era o código-fonte (PHP). O PHP, como é conhecido atualmente, é na verdade o sucessor para um produto chamado PHP/FI. Criada em 1994 por Rasmus Lerdorf, a primeira encarnação do PHP foi um simples conjunto de binários Common Gateway Interface (CGI) escrito em linguagem de programação C (PHP). Em junho de 1995, Rasmus liberou o código-fonte do PHP Tools para o público, o que permitiu que desenvolvedores usassem da forma como desejassem. Isso permitiu – e encorajou – usuários a fornecerem correções para bugs no código e, em geral, aperfeiçoá-lo. Em setembro do mesmo ano, Rasmus expandiu o PHP e – por um breve período – mudou o nome, referindo-se, agora, à ferramenta como FI, abreviação para "Forms Interpreter". A nova implementação incluiu algumas funcionalidades básicas do PHP como bem conhecemos hoje. Tinha variáveis no estilo Perl, interpretação automática de variáveis de formulários e sintaxe HTML embutida (PHP).

(PHP, s.d.)

Essas citações ajudam a entender o contexto e os propósitos iniciais da criação da linguagem. Como veremos a seguir, a primeira finalidade do PHP foi interpretar, do lado servidor, os formulários HTML, fornecendo, assim, dinamismo às páginas Web. Isso porque, com essa linguagem, é possível adicionar recursos como consulta a banco de dados, processamento e tratamento de dados e consumo de recursos externos – como **APIs** –, entre tantas outras possibilidades.

APIs

Application Programming Interface.

Para descrevermos a linguagem PHP, é necessário começar pela sua **sintaxe básica**, apresentando as variáveis, os operadores e as formas de leitura de dados a partir da integração com a HTML.

Atividade 1

Um grupo de desenvolvedores está conversando acerca da linguagem PHP. Analise as sentenças seguintes, que representam as argumentações dos desenvolvedores. Em seguida, marque a opção correta.

A

A linguagem PHP possibilita o uso do paradigma de orientação a objetos, é fracamente tipada, possibilita a geração de páginas web interativas e dinâmicas, a integração com diferentes bancos de dados, entre outros recursos.

B

A linguagem PHP não possibilita o uso do paradigma de orientação a objetos, possibilita apenas o uso do paradigma estruturado, é fracamente tipada, possibilita a geração de páginas web interativas e dinâmicas, a integração com diferentes bancos de dados, entre outros recursos.

C

A linguagem PHP possibilita o uso do paradigma de orientação a objetos, é fracamente tipada, possibilita a geração de páginas web interativas e dinâmicas, mas não possibilita a integração com bancos de dados.

D

A linguagem PHP possibilita o uso do paradigma de orientação a objetos, é fracamente tipada, possibilita a integração com vários bancos de dados, mas não possibilita a geração de páginas web interativas e dinâmicas.

E

A linguagem PHP é a substituta do SQL para desenvolver aplicações de bancos de dados e não tem nenhuma relação com geração de páginas web interativas e dinâmicas.



A alternativa A está correta.

PHP (Hypertext Preprocessor) é uma linguagem de script de propósito geral, amplamente utilizada no desenvolvimento web. PHP é executado no servidor web, gerando HTML, que é enviado para o cliente. Os scripts PHP são interpretados pelo servidor, e não pelo navegador. PHP oferece suporte nativo a uma ampla variedade de bancos de dados, incluindo MySQL, PostgreSQL, Oracle, Microsoft SQL Server, SQLite, entre

outros. Além disso, PHP oferece suporte robusto a POO, permitindo o uso de classes, herança, interfaces e outros conceitos de orientação a objetos.

A sintaxe PHP

Como qualquer linguagem, PHP também tem suas regras de sintaxe, que devem ser obedecidas para o bom funcionamento das aplicações. Além disso, os servidores podem lidar com aplicações implementadas com diferentes tecnologias e, dessa forma, é preciso passar algumas dicas aos servidores para que saibam como proceder. Ou seja, para iniciar um script em PHP, deve-se utilizar a sequência de caracteres `<?php`. Ainda, para finalizar as instruções, utiliza-se ponto e vírgula. Para definir variáveis, utiliza-se o símbolo `$`, sem a necessidade de especificar tipos de dados. A fim de definir uma variável denominada nome, basta fazer o seguinte: `$nome`.

Neste vídeo, exploramos as regras essenciais para escrever e executar scripts em PHP. Descubra como iniciar e finalizar um script; a importância do ponto e vírgula para terminar instruções; como definir variáveis sem especificar tipos de dados. Esse conteúdo é ideal para iniciantes que desejam entender os fundamentos da linguagem e garantir o bom funcionamento de suas aplicações. Não perca!



Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O script PHP deve ser iniciado pela tag `<?php` e fechado com a tag `<?>`. Isso é necessário para que o servidor Web entenda qual código deve ser interpretado e qual deve ser apenas **renderizado**, uma vez que tags HTML podem ser inseridas dentro de um arquivo contendo código PHP. Veja o exemplo a seguir:

Renderizado

Renderização é o processo pelo qual se obtém o produto final de um processamento digital qualquer. Esse processo aplica-se essencialmente em programas de modelagem 2D e 3D, bem como áudio e vídeo.

plain-text

Primeiro código PHP com tags HTML

Título do texto

O código anterior poderia ser salvo como um script PHP.

Por exemplo: `“ola_mundo.php”`. Ao serem interpretados pelo servidor, tanto o código HTML quanto o código PHP, dentro das tags `<?php`, são convertidos em código HTML normal e renderizados no navegador. O servidor Web pode ser configurado para interpretar scripts PHP sem que seja necessário utilizar a extensão `“.php”`. Nesse caso, é usada outra extensão, ou nenhuma – isso é útil quando não queremos revelar a linguagem utilizada em nosso site.

Término de instruções e comentários

As instruções PHP devem ser, obrigatoriamente, **terminadas com a utilização de ponto e vírgula**. Logo, ao final de cada comando, devemos indicar que ele foi terminado.

Em relação aos comentários, temos duas opções:

Opção 1

Os de uma linha são iniciados com duas barras: //



Opção 2

Os de múltiplas linhas são delimitados por /* e */

Veja os exemplos de finalização de comandos e de utilização de comentários no emulador de códigos a seguir. Clique em Executar para verificar a saída do código:



Conteúdo interativo

esse a versão digital para executar o código.

Repare que a tag de fechamento “>” não é obrigatória quando temos apenas código PHP em um script.

Variáveis em PHP

As variáveis são um dos principais recursos em uma linguagem de programação. Em PHP, a definição de uma variável é feita com a utilização do símbolo “\$” **seguido do nome da mesma**. No código de exemplo anterior, a variável “\$var1” foi declarada e, ao mesmo tempo, inicializada.

Em PHP, diferentemente de linguagens como Java, não é necessário informar o tipo de variável. Tal fato concede ao PHP a característica de **linguagem fracamente tipada**. Com isso, não há diferenças no momento da criação de variáveis para receber dados numéricos, textuais, alfanuméricos, entre outros.



Atenção

O único cuidado diz respeito ao momento de atribuição de valores, já que dados do tipo string, por exemplo, precisam ser envolvidos por aspas duplas (“ ”) ou simples (‘ ’).

A respeito dos nomes das variáveis temos, ainda, as seguintes regras:

- Os nomes de variável são case-sensitive. Logo, há diferença entre letras maiúsculas e minúsculas.
- Para ser válido, o nome da variável deve começar com uma letra ou um sublinhado (underscore).
- Após o primeiro caractere (letra ou sublinhado), podem ser utilizadas letras, números e sublinhados.

Atribuição de valores

A atribuição de valores a variáveis em PHP é realizada com a utilização do sinal de igual “=”. Veja este novo exemplo:

php

Repare que diversos tipos de dados foram utilizados e atribuídos às variáveis declaradas. Além disso, diferentes convenções de nomeação foram aplicadas – início com **underscore**; separação por **underscore**; **CamelCase**. É boa prática escolher um único padrão e utilizá-lo em todo o projeto.

CamelCase

É a denominação em inglês para a prática de escrever as palavras compostas ou frases, em que cada palavra é iniciada com maiúsculas e unida sem espaços.



Atenção

Em PHP, as variáveis não inicializadas possuem um valor padrão. Nas do tipo booleano, por exemplo, o valor padrão é false. Logo, é recomendado – e também uma boa prática – inicializar as variáveis antes de utilizá-las, embora isso não seja obrigatório.

Atividade 2

Um desenvolvedor de uma fábrica de software está analisando as características das linguagens de programação antes de decidir qual delas será utilizada para uma aplicação responsável por gerar páginas web dinâmicas. Acerca das características de PHP, marque a resposta correta.

A

PHP é uma linguagem fracamente tipada, de modo que não há regras de sintaxe, tornando-a a opção ideal para aplicações de servidores que gerem páginas web dinamicamente.

B

Os scripts em PHP começam com os caracteres `<?php` e terminam com os caracteres `?>`, ou seja, são características da sintaxe do PHP. Além disso, as instruções são finalizadas com ponto e vírgula.

C

Os scripts em PHP começam com os caracteres `<?php` e terminam com os caracteres `?>`, ou seja, são características da sintaxe do PHP. Além disso, as instruções são finalizadas com ponto de exclamação.

D

Para definir variáveis em scripts PHP, deve-se utilizar a palavra reservada `%var%`. Caso contrário, o script terá erro de compilação, tendo em vista que PHP é uma linguagem compilada.

E

Por ser uma linguagem fracamente tipada, PHP não utiliza o conceito de variáveis como outras linguagens, isto é, os scripts em PHP só trabalham com constantes.



A alternativa B está correta.


A sintaxe do PHP possui diversas regras que ajudam a estruturar e a organizar o código. Um exemplo é o código PHP, delimitado pelas tags `<?php ... ?>`. O conteúdo dentro dessas tags é processado pelo interpretador PHP. Além disso, cada instrução em PHP deve terminar com um ponto e vírgula (;).

Entrada de dados: variáveis de requisição HTTP

Em aplicações que geram páginas web dinâmicas, é comum a utilização de formulários, nos quais os usuários inserem conteúdos a serem enviados aos servidores, também conhecidos como requisições. Os servidores recebem os dados, fazem o devido processamento e, depois, enviam as respostas aos usuários na forma das páginas web. Cada dado que é enviado ao servidor pode ser comparado a variáveis em um programa, isto é, tem nome e conteúdo. A aplicação do servidor consegue extrair os dados baseados nos seus nomes. Além disso, há diferentes métodos de envio dos dados, como o GET e POST. O método GET anexa os dados na URL, enquanto o POST insere os dados no corpo da mensagem.

Neste vídeo, exploramos como aplicações web dinâmicas utilizam formulários para coletar e enviar dados dos usuários aos servidores. Entenda o processo de requisições, em que os servidores recebem, processam e respondem com páginas web. Aprenda sobre os métodos de envio de dados, como GET e POST, e suas diferenças fundamentais.





Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.



Na linguagem PHP estão disponíveis algumas variáveis predefinidas – também chamadas de **superglobais**. Entre elas, estão as de requisição HTTP: **\$_REQUEST**, **\$_POST** e **\$_GET**. Em linhas gerais, essas três variáveis têm a mesma função, ou seja, receber dados provenientes de formulários HTML ou de outras requisições HTTP que façam uso dos métodos POST e GET.



Superglobais

Diversas variáveis predefinidas no PHP são "superglobais", o que significa que elas estão disponíveis em todos os escopos para todo o script. Não há necessidade de fazer global \$variable para acessá-lo dentro de funções ou métodos. Estas variáveis superglobais são: \$GLOBALS.

Métodos de requisição HTTP

A especificação do protocolo HTTP estabelece uma série de métodos de requisição cuja função é indicar qual ação deve ser executada por determinado recurso. Nesse sentido, cada um deles implementa uma diferente semântica. São nove os métodos disponíveis:

- GET
- HEAD
- POST
- PUT
- DELETE
- CONNECT
- OPTIONS
- TRACE
- PATCH

Além dos métodos aqui mencionados, o protocolo HTTP possui, ainda, uma série de outras propriedades relevantes no que tange à programação Web. Recomenda-se, portanto, uma leitura mais aprofundada sobre esse protocolo. A seguir, veremos exemplos de utilização dos métodos GET e POST, que são os mais usados em programas PHP.

Método GET

Este método é utilizado na requisição e na recuperação de recursos de um servidor, como uma página ou um arquivo, entre outros. Veja o exemplo de uma requisição GET:

```
php
```

```
/endereco_servidor/script.php?var1=value1&var2=value&var#=value3
```

Como visto na imagem, a requisição GET é composta pelo endereço (URL e URI) e pela query string – pares de nome/valores (var1=value1, ...).

Em linhas gerais, não deve ser utilizado quando estamos lidando com informações sensíveis, uma vez que a query string fica visível na barra de endereços do navegador. Outra característica importante desse método é que ele pode ser usado a partir de formulários HTML.

Método POST

Este método é usado no envio de dados para o servidor a fim de criar ou atualizar um recurso. A figura seguinte mostra o corpo de uma requisição feita com POST:

```
php
```

```
POST / endereco_servidor/script.php
```

```
Host: dominio.com.br
```

```
var1=value1&var2=value2&var3=value3
```

Assim como o GET, esse método pode ser utilizado em formulários HTML, com a vantagem de não deixar os dados transmitidos visíveis na barra de endereços do navegador – embora seja possível acessá-los analisando a requisição em si.

Variável \$_GET

Em PHP, essa variável predefinida é um array associativo que contém as variáveis recebidas de métodos HTTP GET. Voltando ao exemplo da figura “Exemplo de uma requisição GET”, no script “script.php” as variáveis passadas seriam representadas da seguinte forma:

```
php
```

Como visto anteriormente, no **array \$_GET**, os índices correspondem aos nomes das variáveis da **query string** submetida com o método GET, assim como seus valores.

Variável \$_POST

A exemplo de **\$_GET**, a variável predefinida **\$_POST** também é um **array** associativo. Entretanto, ela contém as variáveis recebidas por meio de métodos POST.

Variável \$_REQUEST

É considerada "curinga", uma vez que exerce múltiplos papéis. Com ela, é possível receber tanto variáveis provenientes de métodos GET quanto POST – e também do método cookies (**\$_COOKIE**).

Cookies

É um fragmento reduzido de dados que fica armazenado no navegador do usuário, proveniente de um servidor Web. São normalmente usados para fins de gerenciamento de sessões, armazenamento de preferências do usuário ou rastreamento.

Sua utilização é semelhante ao que foi visto em **\$_GET** e **\$_POST**.

Atividade 3

Considere que você foi designado para ministrar um treinamento a novos estagiários na empresa de desenvolvimento de software na qual trabalha. Surgiram algumas dúvidas a respeito dos métodos de envio dos dados nas requisições. Alguns estagiários fizeram algumas sugestões, descritas nas opções a seguir. Qual delas é a correta?

A

Com PHP, podemos fazer aplicações que recebam dados provenientes das requisições dos clientes, enviados via método GET, apenas. Além disso, é possível processar e validar esses dados, devolvendo uma resposta aos clientes por meio das páginas web que eles visualizam.

B

Com PHP, podemos fazer aplicações que recebam dados provenientes das requisições dos clientes, enviados via método POST, apenas. Além disso, é possível processar e validar esses dados, devolvendo uma resposta aos clientes por meio das páginas web que eles visualizam.

C

Com PHP, podemos fazer aplicações que recebam dados provenientes das requisições dos clientes, enviados via métodos GET e POST, por exemplo. Além disso, é possível processar e validar esses dados, devolvendo uma resposta aos clientes por meio das páginas web que eles visualizam.

D

Com PHP, não podemos lidar com dados enviados pelos clientes. Nesse caso, devemos recorrer a scripts implementados em linguagem Python.

E

Com PHP, não podemos lidar com dados enviados pelos clientes. Nesse caso, devemos recorrer a scripts implementados em linguagem Java.



A alternativa C está correta.

PHP possibilita lidar com dados provenientes dos clientes, e os dados podem ser enviados por meio de diferentes métodos, como GET e POST, por exemplo.

Operadores PHP

PHP, assim como outras linguagens de programação, possibilita o uso de diferentes operadores, conferindo maior flexibilidade ao desenvolvimento das aplicações. Os operadores aritméticos são os mais intuitivos, tendo em vista que as aplicações podem demandar certos cálculos. Entretanto, existem outros tipos de operadores, como aqueles de atribuição, quando se passam certos conteúdos às variáveis; operadores de comparação que têm a finalidade de fazer a comparação entre variáveis; operadores lógicos, que possibilitam elaborar expressões mais complexas em estruturas de repetição ou condicionais, por exemplo.

Neste vídeo, exploramos como a linguagem PHP utiliza diferentes operadores para aumentar a flexibilidade no desenvolvimento de aplicações. Desde operadores matemáticos, usados em cálculos, até operadores de atribuição, comparação e lógicos, aprenda como cada tipo permite criar expressões complexas e eficientes em seu código. Ideal para desenvolvedores que desejam dominar o uso de operadores em PHP. Assista!



Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.



Um operador – no contexto de linguagens de programação ou mesmo em outras áreas, como na Matemática – tem a função de receber um ou mais valores e resultar em outro valor ou valores. Por exemplo:

$$2 + 2 = 4$$

$$2 - 2 = 0$$

$$2 * 2 = 4$$

$$2 / 2 = 1$$

Os sinais “+”, “-”, “*” e “/” representam as operações matemáticas de adição, subtração, multiplicação e divisão, respectivamente. Logo, são chamados de **operadores aritméticos**. Em PHP, além dos operadores aritméticos, há outros disponíveis. Veremos os principais a seguir.

Operadores aritméticos

Além dos mencionados no exemplo, também estão disponíveis em PHP outros quatro operadores aritméticos, sendo os dois a seguir os mais importantes:

Operador: %

Exemplo de utilização: `$var1 % $var2`

Para que serve: operador de módulo. Retorna o resto da divisão inteira de `$var1` por `$var2`.

Operador: **

Exemplo de utilização: `$var1 ** $var2`

Para que serve: operador exponencial. Retorna o resultado de `$var1` elevado a `$var2`.

Esse tipo de operador é usado, sobretudo, com números (int/integer e float) para a realização de cálculos. Quando utilizado com outra forma de dado, é feita a conversão para o tipo numérico antes que a operação seja realizada.

Operadores de atribuição

São utilizados na atribuição de valores a variáveis. Além de casos simples, como o que vimos com o operador "=", é possível realizar a combinação de operadores de atribuição com os aritméticos. Para ficar mais claro, vamos aos exemplos:

```
php
```

Repare nos comentários inseridos na imagem, onde o resultado de cada combinação é explicado. Para melhor fixação, copie os exemplos e os execute. Tente também alterar os valores ou realizar outras combinações.

Operadores de comparação

São utilizados para comparar dois valores. Veja a seguir os operadores disponíveis e suas funções.

```
==  
$var1 == $var2  
Verifica se $var1 é igual a $var2
```

```
===  
$var1 === $var2  
Verifica se $var1 é idêntica a $var2. Nesse caso, além do valor, verifica se  
ambas são do mesmo tipo
```

```
!=  
$var1 != $var2  
Verifica se $var1 é diferente de $var2
```

<>
\$var1 <> \$var2
Verifica se \$var1 é diferente de \$var2



!=
\$var1 != \$var2
Verifica se não são idênticas/iguais ou se não são do mesmo tipo



<
\$var1 < \$var2
Verifica se \$var1 é menor que \$var2



>
\$var1 > \$var2
Verifica se \$var1 é maior que \$var2




```
<=
$var1 <= $var2
Verifica se $var1 é menor ou igual a $var2
```

```
>=
$var1 >= $var2
Verifica se $var1 é maior ou igual a $var2
```

A partir da versão 7 do PHP, um novo operador foi incluído, o “**spaceship**”, cuja forma de utilização é “**\$var1**↔**\$var2**”. Ele retorna -1, 0 ou 1 quando **\$var1** for, respectivamente, menor, igual ou maior que **\$var2**.

Operadores lógicos

São usados para combinar expressões lógicas. Veja a seguir os operadores lógicos disponíveis em PHP.

```
and
$var1 and $var2
Retorna true se $var1 E $var2 forem verdadeiras
```

or
\$var1 or \$var2
Retorna true se \$var1 OU \$var2 forem verdadeiras



xor
\$var1 xor \$var2
Retorna true se \$var1 OU \$var2 forem verdadeiras, mas não ambas



!
!\$var1
Retorna true se \$var1 não for verdadeira



&&
\$var1 && \$var2
Retorna true se \$var1 E \$var2 forem verdadeiras



```
||
$var1 || $var2
```

Retorna true se \$var1 OU \$var2 forem verdadeiras

Como visto, os operadores **“and” / “&&,” e “or” / “||”** têm a mesma função. Entretanto, “and” e “or” têm maior precedência que seus equivalentes.



Saiba mais

Além dos operadores apresentados, há outros disponíveis em PHP, como os bit a bit (bitwise) e os de controle de error. Na seção Explore+, destacamos um site que contém a lista completa de operadores.

Atividade 4

Suponha que você foi escolhido para ministrar um treinamento a novos estagiários na empresa de desenvolvimento de software na qual trabalha. Surgiram algumas dúvidas a respeito dos operadores utilizados em PHP. Alguns estagiários apontaram algumas sugestões, descritas nas opções seguintes. Dito isso, assinale a alternativa correta.

A

PHP possibilita o desenvolvimento de aplicações que utilizem apenas operadores matemáticos.

B

PHP possibilita o desenvolvimento de aplicações que utilizem apenas operadores lógicos.

C

PHP possibilita o desenvolvimento de aplicações que utilizem apenas operadores de comparação.

D

Com PHP, assim como em outras linguagens, é possível desenvolver aplicações que façam uso de vários operadores, por exemplo, operadores matemáticos, lógicos, de atribuição e de comparação.

E

Com PHP, diferentemente de outras linguagens, não é possível desenvolver aplicações que façam uso de operadores, por exemplo, operadores matemáticos, lógicos, de atribuição e de comparação.



A alternativa D está correta.

Em PHP, os operadores são utilizados para realizar diversas operações em variáveis e valores. Os operadores matemáticos são usados para realizar operações aritméticas; os operadores de atribuição são empregados para atribuir valores a variáveis; já os operadores de comparação são usados para comparar valores. Por fim, os operadores lógicos são utilizados para combinar expressões lógicas.

Lidando com dados provenientes de formulário HTTP


Grande parte das aplicações web lidam com páginas dinâmicas, ou seja, páginas geradas com conteúdos personalizados para cada usuário. Exemplos de aplicações são os mais diversos: e-commerce, bankline, sistemas educacionais, entre tantos outros. Portanto, entender como se implementam aplicações que contenham páginas web, com formulários que enviam os dados ao servidor, que, por sua vez, extrai e os

processa, devolvendo uma resposta ao usuário, é uma tarefa muito importante. Isso constitui o ciclo requisição e resposta, sendo que as aplicações dos servidores esperam por solicitações dos usuários.

Para simplificar, vamos considerar uma aplicação bem simples, que exibe uma mensagem de boas-vindas a um usuário, que enviou seu nome e e-mail em um formulário de uma página web (front-end). A aplicação do servidor (back-end) deve extrair os dados e construir o código da página que será devolvida ao usuário, que a vê no seu navegador. PHP pode realizar essa tarefa de forma bem simples, com poucas linhas de código. Consequentemente, a compreensão sobre como funcionam as requisições e as respostas para uma aplicação simples darão subsídios para aplicações mais complexas.

Neste vídeo, exploramos como as aplicações web lidam com páginas dinâmicas, essenciais em e-commerce, sistemas bancários e educacionais. Você aprenderá sobre o ciclo de requisição e resposta. Usando um exemplo simples em PHP, mostraremos como capturar dados de um formulário e exibir uma mensagem de boas-vindas personalizada, preparando você para criar aplicações web mais complexas.





Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.



Roteiro de prática

Considere uma aplicação bem simples, que exibe uma mensagem de boas-vindas a um usuário, que enviou seu nome e e-mail em um formulário de uma página web (front-end). A aplicação do servidor (back-end) deve extrair os dados e construir o código da página que será devolvida ao usuário, que a vê no seu navegador.

A referência para este exemplo é o W3schools (seção PHP do site).

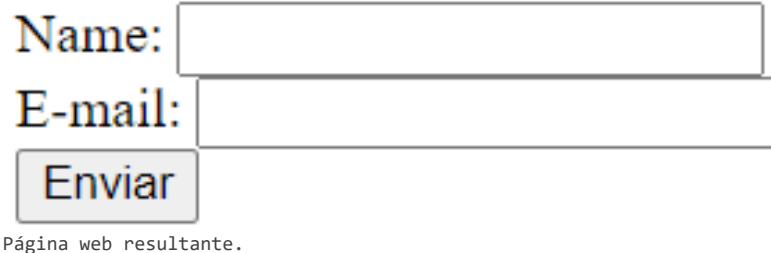
Inicialmente, vamos considerar a página HTML que contém o formulário com dois campos: nome e e-mail.

```
plain-text
```

Name:

E-mail:

A página resultante do código anterior é a seguinte:



The image shows a simple web form. It has two text input fields. The first is labeled 'Name:' and the second is labeled 'E-mail:'. Below these fields is a button labeled 'Enviar'. The form is styled with a light blue background and a thin border.

Página web resultante.

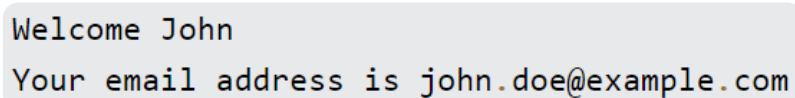
Quando o usuário preencher os campos do formulário e clicar no botão, os dados serão enviados ao servidor, por meio do método POST, para serem processados pela aplicação `welcome.php`, que pode conter linhas de código conforme mostrado a seguir.

```
plain-text
```

Welcome

Your email address is:

E a página resultante, que o usuário visualiza no navegador, pode ser semelhante à seguinte imagem. Observe!



The image shows a response message. It has a light blue background and a thin border. The text inside is 'Welcome John' followed by 'Your email address is john.doe@example.com'.

Exemplo da resposta.

Vamos detalhar um pouco mais as linhas de código da imagem de exemplo de aplicação para welcome.php , começando pela linha de código mostrada a seguir.

```
<?php echo $_POST["name"];
```

Obtendo o dado name da requisição.

O código `$_POST["name"]` extrai o conteúdo do campo do formulário, denominado name, que foi enviado na requisição por meio do método POST.

Agora, vamos analisar a próxima linha de código.

```
<?php echo $_POST["email"];
```

Obtendo o dado email da requisição.

De forma semelhante, o código `$_POST["email"]` extrai o conteúdo do campo do formulário, denominado email, que também foi enviado na requisição por meio do método POST.

Atividade 5

A

Deve-se garantir que o atributo action do formulário se refira à criação do campo de senha, de modo que, ao digitar, os caracteres são mascarados por pontos, objetivando a segurança dos usuários.

B

Para que a aplicação funcione adequadamente, o front-end deve enviar os dados via método GET e o back-end deve utilizar o método POST.

C

O desenvolvedor deve utilizar a função `file_get_contents()` para acessar os dados enviados na requisição, ou seja, acessar os dados digitados no formulário por parte do usuário.

D

O desenvolvedor deve usar o método `isset()` para verificar se os dados do formulário foram recebidos corretamente no arquivo PHP. Ele deve envolver o acesso aos dados do formulário com `isset()`.

E

O desenvolvedor deve utilizar o método POST tanto no front quanto no back-end. Além disso, outros cuidados envolvem a definição correta dos nomes dos campos de formulário. Ele deve garantir que os nomes dos campos no HTML correspondam aos nomes usados para acessar os dados no arquivo PHP.



A alternativa E está correta.

Para que os dados do formulário sejam recebidos corretamente no código PHP, deve-se garantir que os nomes dos campos no HTML correspondam aos nomes usados para acessar os dados no código PHP. Isto é, os nomes dos campos no HTML devem corresponder exatamente aos índices usados para acessar os dados na matriz `$_POST[]` do código PHP. Isso garante que os dados do formulário sejam transmitidos corretamente para o PHP.

2. Estruturas de decisão e de repetição

Condicionais: estruturas de decisão

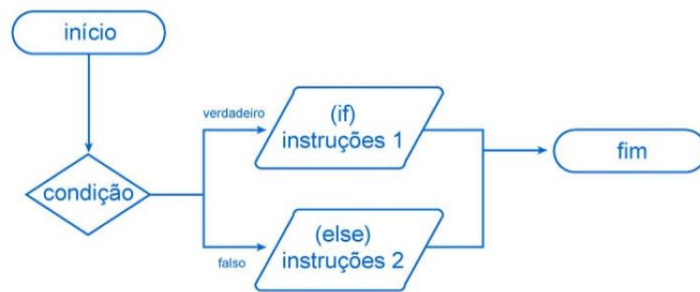
Com frequência, há situações em que o mesmo programa pode seguir caminhos diferentes dependendo de certas condições, o que reflete situações cotidianas, por exemplo, se a idade for até 11 anos, trata-se de criança. Se for de 12 até 18 anos, é adolescente. Para traduzir essas condições para os programas, há certas instruções específicas denominadas condicionais ou estruturas de decisão. São estruturas bastante úteis e versáteis, utilizadas em programas desenvolvidos por meio das mais diversas linguagens, como PHP. Portanto, o entendimento dessas estruturas, assim como as respectivas sintaxes, é bastante importante para o desenvolvimento de aplicações.

Neste vídeo, explicamos como programas podem seguir diferentes caminhos com base em certas condições, refletindo situações cotidianas como classificar idade em criança ou adolescente. Exploramos as instruções condicionais ou estruturas de decisão, essenciais em várias linguagens de programação como PHP. Aprenda neste vídeo a aplicar essas estruturas versáteis e fundamentais para o desenvolvimento de aplicações.



Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.



Fluxo das estruturas de decisão.

A figura anterior demonstra o fluxo de uma **estrutura de decisão**. Nela é possível ver que, a partir da verificação de uma condição, o programa se divide em dois caminhos possíveis. Esse é um exemplo simples, uma vez que várias condições podem ser verificadas ao mesmo tempo, assim como vários caminhos alternativos podem ser seguidos.

if

As **estruturas de decisão e de repetição** são recursos importantes em uma linguagem de programação. Com elas, é possível mudar o fluxo de um programa por meio de verificações (estruturas de decisão) e também executar diversas vezes partes do programa (estruturas de repetição). A seguir, veremos como utilizá-las em PHP.

A sintaxe da estrutura de controle **if** em PHP é composta pela condição (ou condições) a ser verificada e, caso seja verdadeira, é seguida da instrução (ou instruções) a ser executada. Logo, temos que as condições são avaliadas por seus valores booleanos, isto é, se são verdadeiras ou falsas. Vejamos este fragmento de código PHP a seguir:

```
php
    $var2){
        echo "$var1 é maior que $var2";
    }
```

No exemplo foi realizada apenas uma verificação – a comparação entre as duas variáveis definidas. Caso a primeira seja maior que a segunda, uma mensagem é exibida na tela. É possível também realizar outras verificações em um mesmo **if**. Para isso, basta utilizar subgrupos.

Antes de prosseguirmos, cabe destacar mais alguns elementos da sintaxe do **if**:

- A condição (ou condições) a ser avaliada deve ser envolvida por parênteses, sendo possível incluir subgrupos dentro de novos parênteses.
- Múltiplas instruções devem ser envolvidas com chaves. No exemplo, como só há uma instrução a ser executada, as chaves são opcionais e podem ser omitidas.

Else

Para apresentar a estrutura **else**, voltaremos ao exemplo anterior, no qual é verificado se uma variável era maior do que a outra e, em caso positivo, exibida uma mensagem.

O que acontece em nosso programa caso a condição em questão não seja verdadeira? Qual retorno é exibido nesse caso?

Digite o código no emulador seguinte, execute-o e veja você mesmo:

Como \$var1 é menor que \$var2, nada é exibido no navegador, já que a condição é falsa. Nessa situação, para imprimirmos uma mensagem informando que a condição é falsa, ou seja, que \$var1 é menor que \$var2, faremos uso do **else**. Veja como ficaria nosso código nesse ponto:

Conteúdo interativo

esse a versão digital para executar o código.

Como demonstrado no exemplo, a estrutura `else` tem a função de definir um fluxo alternativo ao nosso programa, caso uma determinada condição seja falsa.

Em relação à sua sintaxe, vale o que foi dito para if, sobre múltiplas instruções precisarem ser envolvidas em chaves.

Elseif/else if

É uma combinação das duas instruções vistas. Logo, sua função é definir fluxo (ou fluxos) alternativo caso uma condição verificada com **if** seja falsa. Entretanto, ela permite ainda que uma nova condição (ou até mesmo condições) seja avaliada. Vamos ao exemplo de seu uso:



Conteúdo interativo

esse a versão digital para executar o código.



Repare que além da primeira verificação, com o **if**, foi inserida uma segunda, com **elseif**. Ao final, a instrução **else** representa o fluxo caso nem a condição do **if** nem a do **elseif** sejam verdadeiras.

Sobre sua sintaxe, além do que já foi dito no **if**, cabe destacar que não há limites de instruções **elseif** dentro de uma declaração **if**.

Switch

Pode ser comparada a uma série de instruções **if**, possuindo uma sintaxe um pouco diferente e usada sobretudo quando se deseja verificar diferentes valores, inúmeras vezes, em uma mesma variável. Vejamos sua sintaxe por meio de um novo exemplo:



Conteúdo interativo

esse a versão digital para executar o código.



Com o **switch** temos uma série de verificações e, ao final, uma instrução padrão (**default**) a ser executada, caso nenhuma das condições seja verdadeira. Para fins de prática, modifique o valor da variável **\$var1** no emulador anterior, execute o código e veja o resultado obtido.

Formas alternativas

PHP permite que sejam utilizadas formas alternativas das instruções vistas. Em linhas gerais, troca-se a chave de abertura por dois pontos e a de fechamento pela palavra reservada **"end"** seguida do nome da instrução. Veja o exemplo:



Conteúdo interativo

esse a versão digital para executar o código.



Essa sintaxe é muito utilizada quando misturamos código HTML e PHP.

Outra sintaxe alternativa interessante presente no PHP é o **operador ternário**. Por meio dele é possível avaliar uma condição e atribuir um valor de acordo com a validação. Veja o exemplo para ficar mais claro:



Conteúdo interativo

esse a versão digital para executar o código.



Note que uma condição foi verificada dentro de parênteses. Caso verdadeira, após o sinal “?” é atribuído o valor “11”. Caso negativa, após o sinal “:” é atribuído o valor “9”. Execute novamente o código no emulador acima, mas antes modifique o valor da variável **\$var1**. Analise o resultado obtido.

Atividade 1

Considere que você foi convidado para participar de um podcast da empresa na qual trabalha. Uma das perguntas que foram feitas se refere aos recursos de programação comumente utilizados em diversas linguagens, como estruturas condicionais. Surgiram várias possíveis respostas a essa pergunta, descritas nas opções seguintes. Qual delas é a correta?

A

Em códigos desenvolvidos com PHP, podemos utilizar as estruturas condicionais para definir caminhos distintos que o programa pode seguir, ou seja, trechos de códigos que serão executados dependendo das condições estabelecidas.

B

PHP é uma linguagem que não possibilita o uso de estruturas condicionais e deve-se optar por usar JavaScript nessas situações, de forma a complementar a lacuna existente com PHP.

C

Os programas computacionais, em geral, não são implementados com a possibilidade de utilização de condicionais, tendo em vista que isso fere o preceito de complexidade de algoritmos.

D

PHP só possibilita o uso de estruturas de repetição, que podem substituir as estruturas condicionais.

E

Em códigos PHP, o desenvolvedor deve escolher se utiliza estruturas condicionais ou de repetição, uma vez que não é possível utilizar as duas modalidades no mesmo código.



A alternativa A está correta.

Em códigos desenvolvidos com PHP, assim como em outras linguagens, é possível utilizar estruturas condicionais, por serem recursos básicos no desenvolvimento de aplicações computacionais.

Laços: estruturas de repetição

As estruturas de repetição também são bastante utilizadas independentemente do paradigma adotado e de linguagens de programação. Proporcionam a redução da quantidade de linhas de código, pois, dentro dessas estruturas, são inseridas as linhas de código que vão se repetir determinado número de vezes, de modo que o código fique mais sucinto, facilitando a leitura e o entendimento. PHP suporta diferentes estruturas de repetição, a saber: `while`, `do-while`, `for` e `foreach`. Cada uma delas tem suas particularidades, que devem ser observadas para o bom funcionamento dos programas. Podem ser utilizadas mais de uma vez e combinadas em um mesmo código.

Descubra, neste vídeo, a importância das estruturas de repetição, independentemente do paradigma adotado e da linguagem de programação utilizada. Saiba como essas estruturas podem reduzir a quantidade de linhas de código, tornando-o mais conciso e fácil de entender. Em PHP, você encontrará diferentes opções, como `while`, `do-while`, `for` e `foreach`, cada uma com suas particularidades. Aprenda a utilizá-las de forma eficiente e a combiná-las para obter o melhor desempenho em seus programas.



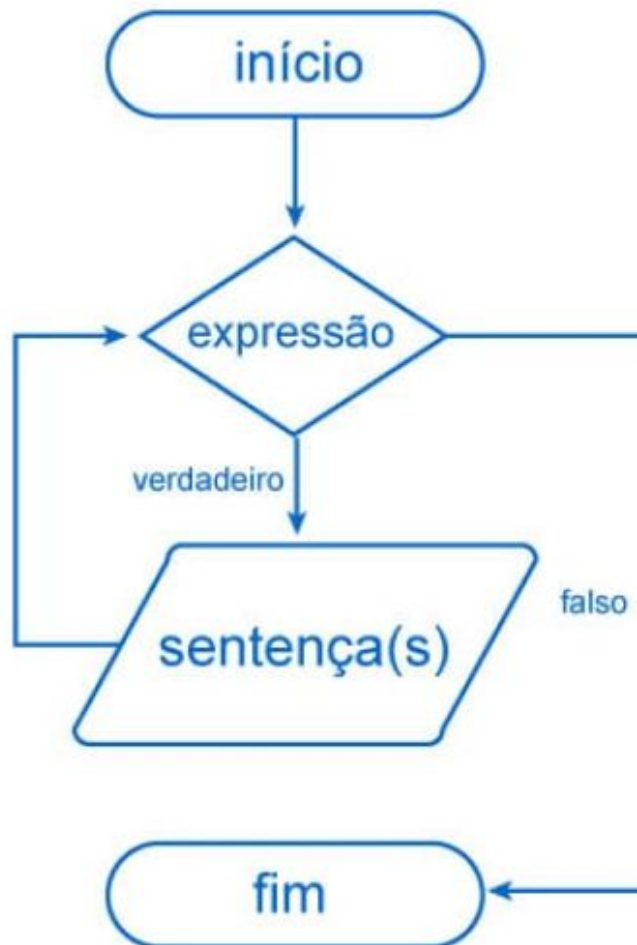
Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.



As **estruturas de repetição**, muitas vezes também chamadas de **laços**, permitem que instruções ou trechos de códigos sejam executados de forma repetitiva. Sua sintaxe define as condições ou expressões que devem ser verificadas e, caso essas sejam verdadeiras, quais instruções devem ser executadas e por quantas vezes. A próxima figura apresenta o fluxo básico das estruturas de repetição.

Em PHP estão disponíveis as seguintes estruturas: **`while`**, **`do-while`**, **`for`**, **`foreach`**. A seguir, descreveremos cada uma delas.



Fluxo das estruturas de repetição.

While

O laço While possui uma sintaxe simples: enquanto uma expressão for verdadeira, uma série de instruções será executada de forma repetida. Para imprimirmos na tela os números de 2 a 20, pulando de 2 em 2, poderíamos utilizar o seguinte código:

■



Conteúdo interativo

esse a versão digital para executar o código.

Repare que, no exemplo foram mostradas a forma normal e a forma alternativa do laço **while** – também presente nas demais estruturas de repetição em PHP. Para praticar, tente alterar a condição no emulador anterior e veja o resultado, executando o código novamente.

O comando **echo "\n"** é utilizado para imprimir uma linha em branco quando o script é executado via linha de comando.

Do-while

Esse laço é semelhante ao anterior, exceto pelo fato de que a verificação, aqui, é feita ao final. Com isso, todas as instruções do laço serão executadas pelo menos uma vez. Veja o exemplo:



Conteúdo interativo

esse a versão digital para executar o código.

For

Esse laço possui sintaxe um pouco diferente do que vimos nos anteriores. Vamos ao exemplo:



Conteúdo interativo

esse a versão digital para executar o código.

Com o laço **for**, temos três expressões sendo avaliadas. Considerando o exemplo anterior, temos:

- A primeira expressão – “`$i = 1`” – é avaliada, incondicionalmente, no início da repetição.
- A seguir, a cada interação, a segunda expressão – “`$i <= 20`” – é avaliada. Caso seja verdadeira, o loop continuará.
- Por último, ao final de cada interação, a terceira expressão – “`$i++`” – é avaliada e executada.

Outra possibilidade nesse laço é avaliar **múltiplas expressões**, que deverão ser separadas por vírgulas. Além disso, também é possível inserir **expressões vazias**.

Foreach

A última estrutura de repetição que veremos em PHP é a **foreach**. Esse laço é parecido com o **for**, possuindo uma sintaxe mais simples e sendo muito propício para realizar interações em **arrays**. Veja o exemplo:



Conteúdo interativo

esse a versão digital para executar o código.



Para fins de comparação e demonstração da diferença entre ambos, em nosso exemplo a mesma operação – imprimir os nomes dos carros – foi realizada tanto com o laço **foreach** quanto com o laço **for**.





Saiba mais

Além dessas estruturas, em PHP estão disponíveis outros comandos relacionados às estruturas de repetição, como o break e o continue.

Atividade 2

Considere que você foi convidado para participar de um podcast da empresa na qual trabalha. Uma das perguntas que foram feitas se refere aos recursos de programação comumente utilizados em diversas linguagens, como estruturas de repetição. Surgiram várias possíveis respostas a essa pergunta, descritas nas opções seguintes. Qual delas é a correta?

A

As estruturas de repetição, embora sejam recursos interessantes para a otimização da quantidade de linhas de código, não são suportadas por PHP, por se tratar de linguagem de scripts, aplicada no front-end de aplicações web.

B

As estruturas de repetição são comumente utilizadas nos programas desenvolvidos nas mais diversas linguagens de programação. Entretanto, PHP suporta apenas um tipo de estrutura de repetição, denominada for.

C

As estruturas de repetição são comumente utilizadas nos programas desenvolvidos nas mais diversas linguagens de programação. E PHP suporta diferentes tipos de estruturas de repetição, por exemplo, `for`, `while`, `do-while` e `foreach`.

D

As estruturas de repetição são comumente utilizadas nos programas desenvolvidos nas mais diversas linguagens de programação. Entretanto, PHP suporta apenas um tipo de estrutura de repetição, denominada `while`.

E

As estruturas de repetição, embora sejam recursos interessantes para a otimização da quantidade de linhas de código, não são suportadas por PHP, por se tratar de linguagem de scripts, aplicada no back-end de aplicações web.



A alternativa C está correta.

Por se tratar de recurso utilizado com frequência nos programas desenvolvidos nas mais diversas linguagens, as estruturas de repetição são suportadas por PHP, que contempla diferentes tipos, a saber: `for`, `while`, `do-while` e `foreach`.


Usando laços e estruturas de decisão na prática

Considere uma aplicação que calcule a média das notas de prova dos alunos de uma turma, ou seja, deve-se somar todas as notas e dividir o valor resultante pela quantidade de alunos. Se essas notas estiverem em um array, basta percorrer os elementos do array, calculando a soma total das notas. Com PHP, há estruturas prontas para facilitar trabalhos repetitivos como o de percorrer os elementos de um array, poupando tempo dos desenvolvedores e otimizando a quantidade de linhas finais no código. Se não fossem essas estruturas de repetição, teríamos mais trabalho para implementar programas que possam realizar a tarefa descrita.

E se, além do cálculo da média da turma, fosse necessário exibir uma mensagem ou realizar alguma ação de acordo com o resultado da média? Como fazer isso? Com as estruturas de decisão, é simples resolver esse problema. Basta vincular as ações que serão realizadas por meio de linhas de código inseridas dentro de cada condição possível. Tanto as estruturas de repetição quanto aquelas de decisão são fundamentais para o bom desenvolvimento de programas e, conseqüentemente, o entendimento de como utilizá-las é requisito necessário aos desenvolvedores de software.

A partir deste vídeo, imagine que você, desenvolvedor de software, precisa calcular a média das notas de uma turma de alunos. Parece complicado, não é mesmo? Mas com PHP, tudo fica mais fácil! Com estruturas prontas para percorrer arrays, você economiza tempo e linhas de código. E se precisar tomar decisões com base na média, também é moleza! As estruturas de decisão estão aí para ajudar. Então, já sabe: dominar essas ferramentas é essencial para o sucesso no mundo da programação.





Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.



Roteiro de prática

Considere uma turma com dez alunos. Eles fizeram uma prova e as notas já foram lançadas, pelo professor, no sistema computacional da instituição de ensino. Para simplificar, vamos imaginar que essas notas estejam em um array. Inicialmente, deve-se calcular a média da turma, depois, exibir alguma mensagem de acordo com o resultado final da média. O código, mostrado a seguir, ilustra como é possível resolver esse problema. Observe!

plain-text

Inicialmente, o vetor \$notas contém as notas dos dez alunos da turma.

```
$notas=[6.6, 8.4, 9.2, 7.8, 5.1, 7.9, 8.3, 7.7, 6.4, 8.6];
```

Definição do array \$notas.

Na sequência, a estrutura de repetição foreach percorre os elementos do array \$notas, atribuindo o valor de cada elemento à variável \$i. Dentro do bloco do foreach, a cada iteração, a variável \$soma adiciona o valor correspondente da variável \$i, conforme mostra o trecho de código a seguir.

```
foreach ($notas as $i) {  
    $soma += $i;  
}
```

Uso do foreach.

Depois, a média das notas é calculada.

```
$media = $soma / 10;
```

Cálculo da média.

Em seguida, dependendo do valor da média, é mostrada uma mensagem. Para isso, foi utilizada uma estrutura de decisão, conforme mostrado na imagem que segue.

```
if($media < 6.0) {  
    echo "Média da turma = " . $media. ". Estude mais!";  
}  
elseif ($media == 6.0) {  
    echo "Média da turma = " . $media. ". Na média!";  
}  
else {  
    echo "Média da turma = " . $media. ". Acima da média!";  
}  
}
```

Uso da estrutura de decisão.

Atividade 3

Imagine que você está participando de um processo seletivo para estágio e, em uma das etapas, você deve analisar o código mostrado a seguir. Depois, marque a opção que descreva a saída do programa, isto é, o que acontece ao executar o código.

php

A

2 4 6 8 10

B

1 2 3 4 5 6 7 8 9 10

C

1 3 5 7 9

D

2 4 6 8

E

2 3 4



A alternativa A está correta.

Trata-se de um código PHP que exibe na tela somente os números pares dentro do intervalo de 1 a 10.

Linha 1: A tag `<?php` inicia o bloco de código PHP.

Linha 2: O laço `for` é usado para iterar de 1 a 10. A variável `$i` começa em 1 e é incrementada em 1 a cada iteração até chegar a 10.

Linha 3: A estrutura `if` verifica se o valor de `$i` é par, usando o operador de módulo (%). Se `$i % 2` é igual a 0, então `$i` é par.

Linha 4: Se a condição `for` verdadeira (ou seja, se `$i` for par), o valor de `$i` é impresso seguido de um espaço (" ").

Linha 5: A tag `?>` encerra o bloco de código PHP.

3. Vetores e funções

Arrays: vetores

Os arrays são recursos muito utilizados no desenvolvimento de sistemas computacionais e podem ser uni ou multidimensionais. No caso dos arrays unidimensionais, podemos compará-los a vetores, em que é possível armazenar vários conteúdos em uma única variável de apenas uma dimensão. E cada elemento tem seu respectivo índice, que possibilita referenciá-lo facilmente. Em PHP, assim como em várias outras linguagens, podemos utilizar os arrays para armazenar dados numéricos e textuais. Podemos utilizar um array para armazenar as notas dos alunos de uma turma. Podemos também ter um array com os nomes dos alunos, ou seja, string de caracteres.

Descubra neste vídeo como os arrays podem facilitar o desenvolvimento de sistemas computacionais. Entenda a diferença entre arrays uni e multidimensionais, e aprenda como utilizá-los em PHP e outras linguagens. Saiba como armazenar notas e nomes de alunos em arrays de forma prática e eficiente. Assista ao vídeo e domine essa poderosa ferramenta de programação!

Conteúdo interativo



Acesse a versão digital para assistir ao vídeo.

Os vetores, ou **arrays**, são variáveis que armazenam um grupo de itens relacionados. Observando o exemplo utilizado ao final do último módulo, vimos a variável “**\$carros**” armazenar nomes de carros.

Os **arrays** podem ser vistos, numa abstração com o nosso dia a dia, como listas escritas em uma folha: nela inserimos vários itens, de forma ordenada. Com isso, cada novo elemento é incluído ao final da lista – embora seja possível inseri-los também em outra ordem. Nas linguagens de programação em geral, e especificamente em PHP, os **arrays** funcionam exatamente desta forma: uma lista ordenada na qual novos itens podem ser inseridos, assim como os existentes podem ser deletados ou substituídos.

Os **arrays** são compostos por dois elementos principais: o item em si e o seu índice, que é a posição que este ocupa dentro de um **array**. Esse número se inicia em 0. Entretanto, os índices também podem ser formados por strings.

Em PHP estão presentes diferentes tipos de **arrays**. São eles:

array numérico
Índice composto por números inteiros.

array associativo
Índice composto por strings.

array misto
Índices numéricos e associativos.

O mesmo vale para os elementos de um vetor: podemos tanto ter **strings** quanto números, entre outros tipos de dados, como **arrays** de **arrays**.

Criação de arrays e atribuição de valores

A linguagem PHP, por ser simples e bastante flexível, permite diferentes formas de declarar e atribuir valores em um **array**. Não existe uma maneira melhor do que a outra – qualquer uma delas pode ser utilizada. Nesse sentido, cabe destacar a última forma: ela é bastante útil em situações nas quais não sabemos o tamanho do **array** ou a quantidade de itens que ele receberá.

Ao estudar os exemplos, repare, ainda, nos comentários inseridos no código. Vamos a eles:

```
php
```

Os **arrays** anteriores são **numéricos**. Vejamos outros exemplos, agora com vetores **associativos**. Repare que a principal diferença é a utilização de **strings** no lugar de números para definir os seus índices.

```
php
"Fusca",
    'chevrolet' => "Monza",
    'fiat'      => "Tempra"
);

//Segunda forma
$carros = [
    'vw'        => "Fusca",
    'chevrolet' => "Monza",
    'fiat'      => "Tempra"
];

//Terceira forma
$carros['vw']      = "Fusca";
$carros['chevrolet'] = "Monza";
$carros['fiat']    = "Tempra";
```

Por fim, veremos exemplos de **array** com índices **numéricos e associativos**. Repare que a sintaxe é parecida com a vista na declaração dos associativos, ou seja, cada par “**índice/valor**” é separado por ‘**=>**’. Vamos ao código:



Conteúdo interativo

esse a versão digital para executar o código.

Vamos praticar mais um pouco! No código do emulador anterior, insira a seguinte linha em seu final:

```
print_r($carros);
```

Após isso, execute o código novamente e veja que o array será impresso no campo INPUT do emulador.

As formas vistas nos exemplos anteriores são as mais simples para a criação e inserção de elementos. Entretanto, a linguagem oferece outras formas, por meio do uso de funções como a **array_push** (que adiciona elementos ao final de um **array**) e a **array_unshift** (adiciona elementos no início de um **array**).



Saiba mais

Além dessas, há funções que permitem gerar novos vetores, combinar vetores existentes e muito mais. Pesquise nos sites indicados no Explore+.

Remoção de elementos de um array

Há algumas formas de remover elementos de um **array**.

Primeira forma

A primeira é definindo o valor do elemento como vazio. Nesse caso, embora o valor do elemento seja removido, o seu índice permanece no vetor, que também mantém o seu tamanho inicial.

Segunda forma

Outra forma é fazendo uso de duas funções: **unset** e **array_splice**.

O código a seguir – que como os demais é totalmente funcional e, portanto, deverá ser executado por você – contém exemplos de utilização das formas apresentadas. Seguem algumas observações sobre o código e o uso das funções:

"print_r"

A função "**print_r**" imprime os elementos de um **array**.

"count"

A função "**count**" retorna o tamanho (quantidade de elementos) de um **array**. Essa função, inclusive, é muito útil quando trabalhamos com vetores.

"unset"

A função "**unset**" recebe como parâmetro o **array** e índice ou índices que desejamos remover. Além disso, é possível também remover o vetor inteiro, passando-o como parâmetro e sem definir nenhum índice.

"array_splice"

A função "**array_splice**" recebe como parâmetros o **array** a ser manipulado, o **offset** (índice a partir do qual desejamos excluir elementos) e o **length** (quantidade de itens que queremos excluir).

php

```
"Fusca",
    0      => "Passat",
    'chevrolet' => "Monza",
    1      => "Chevette",
    'fiat'    => "Tempra",
    2      => "Uno"
);
print_r($carros);
echo "O tamanho atual do array é: " . count($carros);
echo "\n\n";
//Definindo o valor do índice 0 como vazio
$carros[0] = '';

print_r($carros);
echo "O tamanho atual do array é: " . count($carros);
echo "\n\n";

//Removendo dois elementos do array com unset
unset($carros['fiat'], $carros[1]);

print_r($carros);
echo "O tamanho atual do array é: " . count($carros);
echo "\n\n";

//Removendo elementos do array com array_splice
array_splice($carros, 1,2);

print_r($carros);
echo "O tamanho atual do array é: " . count($carros);
```

Arrays multidimensionais

O array que será visto no exemplo a seguir é multidimensional. Ou seja, ele é composto por mais de uma dimensão – nesse caso, duas. Olhando para o exemplo, temos:

php

```
$frutas = array (
    "vermelhas" => array(
        "melancia",
        "cereja",
        "framboesa",
        "morango"
    ),

    "citricas" => array(
        "laranja",
        "limao",
        "abacaxi",
        "mexerica"
    ),
);
```

- Dois índices principais, associativos, cujas chaves são 'vermelhas' e 'cítricas'.
- Cada uma dessas chaves possui um novo **array** numérico, que contém quatro elementos.

Atividade 1

A

Os arrays podem armazenar vários conteúdos em uma única variável. Em PHP, podemos utilizá-los para armazenar dados numéricos ou textuais. Cada elemento do array tem um índice que permite referenciá-lo.

B

Os arrays podem armazenar vários conteúdos em variáveis separadas, ou seja, cada elemento do array é uma variável independente, com um índice que permite referenciá-los.

C

Os arrays são estruturas muito utilizadas no desenvolvimento de sistemas, mas PHP não suporta a criação e a manipulação de arrays, sendo necessário importar bibliotecas de JavaScript para esse propósito.

D

PHP suporta apenas arrays unidimensionais, que armazenem dados numéricos.

E

PHP suporta apenas arrays unidimensionais, que armazenem dados textuais, ou seja, string de caracteres.



A alternativa A está correta.

Com PHP, podemos escrever códigos que utilizem arrays, que são estruturas que podem armazenar vários conteúdos na mesma variável, sejam eles numéricos, sejam textuais. Além disso, cada elemento do array é referenciado por um índice.

Demonstração prática do uso de array multidimensional em PHP

O uso de arrays, uni ou multidimensionais, é muito importante no desenvolvimento de software e, consequentemente, entender como implementá-los é tarefa essencial para os desenvolvedores. Considere a situação em que os alunos de uma turma fazem duas provas no semestre letivo, P1 e P2. Isto é, ao lado dos nomes dos alunos, devem aparecer as notas de provas, conforme mostrado a seguir.

Nesse contexto, a utilização de array multidimensional é mais indicada para armazenar os nomes e as notas dos alunos. Podemos considerar o array multidimensional um agrupamento de arrays unidimensionais, que armazenam o nome e as notas de um único aluno.

Assim, o primeiro array unidimensional terá os seguintes elementos:


O segundo array unidimensional terá os seguintes elementos:

E assim por diante. Os arrays unidimensionais são agrupados e fazem parte do array externo que compreende todos eles, formando o array multidimensional.

Nesse exemplo, o array multidimensional tem duas dimensões dadas por linhas e colunas. Logo, para referenciar algum elemento, precisamos de dois índices: um para referenciar a linha e outro para referenciar a coluna do elemento.

No campo do desenvolvimento de software, a utilização de arrays é essencial, especialmente quando se trata de armazenar informações como notas de alunos. Neste vídeo, você verá uma situação em que cada aluno possui notas de P1 e P2. Os arrays multidimensionais se destacam como a melhor opção para organizar os dados de maneira eficiente. Essas estruturas permitem agrupar os dados de maneira que seja possível armazenar, além dos nomes dos alunos, suas respectivas notas.





Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.



Roteiro de prática 1

Considere a situação em que os alunos de uma turma fazem duas provas no semestre letivo: P1 e P2. Ou seja, ao lado dos nomes dos alunos, devem aparecer as notas de provas, conforme mostrado a seguir.

Como implementar uma aplicação em PHP que possa utilizar um array multidimensional para mostrar os nomes e as notas dos alunos, além de mostrá-las ao usuário?

Inicialmente, vamos mostrar todo o código e, depois, vamos analisá-lo mais detidamente.

plain-text

```
";  
echo $alunos[1][0].": P1: ".$alunos[1][1].", P2: ".$alunos[1][2]."  
";  
echo $alunos[2][0].": P1: ".$alunos[2][1].", P2: ".$alunos[2][2]."  
";  
echo $alunos[3][0].": P1: ".$alunos[3][1].", P2: ".$alunos[3][2]."  
";  
?>
```

O código das linhas 6 a 11 dessa imagem mostra a criação do array multidimensional \$alunos, contendo quatro arrays unidimensionais. Cada um desses arrays tem o nome e as notas de P1 e P2 dos alunos, conforme destacado na próxima imagem.

```
5 <?php  
6 $alunos = array (  
7     array("João", 8.7, 9.4),  
8     array("Maria", 9.2, 8.9),  
9     array("Luis", 7.8, 8.4),  
10    array("Fernanda", 8.7, 9.1)  
11 );
```

Criação de array multidimensional.

Em seguida, os elementos de cada array interno são exibidos na tela por meio da referência de linha e coluna a que pertencem, utilizando dois pares de colchetes. O primeiro par de colchete refere-se ao índice da linha, e o segundo, ao da coluna, como mostrado em seguida.

```
13 echo $alunos[0][0].": P1:".$alunos[0][1].", P2: ".$alunos[0][2]."<br>;  
14 echo $alunos[1][0].": P1: ".$alunos[1][1].", P2: ".$alunos[1][2]."<br>;  
15 echo $alunos[2][0].": P1: ".$alunos[2][1].", P2: ".$alunos[2][2]."<br>;  
16 echo $alunos[3][0].": P1: ".$alunos[3][1].", P2: ".$alunos[3][2]."<br>;
```

Manipulação dos elementos do array.

Atividade 2

Um desenvolvedor está trabalhando em um projeto PHP que envolve a manipulação de um array contendo informações sobre produtos. Durante o desenvolvimento, surge a necessidade de remover um produto específico do array, garantindo que o array permaneça íntegro e que seu tamanho seja ajustado conforme necessário.

Qual ação o desenvolvedor deve realizar para alcançar esse objetivo?

A

Utilizar a função `print_r()` para visualizar os elementos do array e, em seguida, selecionar manualmente o produto a ser removido.

B

Utilizar a função `count()` para determinar o tamanho do array e, em seguida, deletar o produto desejado.

C

Utilizar a função `unset()` e especificar o índice do produto que deseja remover.

D

Utilizar a função `array_splice()` e especificar o índice do produto que deseja remover.

E

Utilizar a função `array_unshift()` para remover o primeiro produto do array.



A alternativa D está correta.

Para remover um produto específico de um array em PHP de forma prática e garantir que o array permaneça íntegro, o desenvolvedor pode utilizar a função `array_splice()` e especificar o índice do produto que deseja remover. Essa função permite remover elementos de um array e ajustar o array conforme

necessário, mantendo sua integridade. As outras opções mencionam abordagens que não estão diretamente relacionadas à remoção prática de produtos específicos de um array.

Funções em PHP

O uso de funções é muito comum no desenvolvimento de software e está relacionado ao conceito de modularização, isto é, à divisão de um problema maior e mais complexo em partes menores, denominadas módulos, de mais fácil entendimento e implementação. PHP, assim como outras linguagens de programação, tem várias funções prontas, denominadas funções nativas, que podem ser utilizadas diversas vezes nos programas para facilitar o trabalho de desenvolvimento de software. Entretanto, os desenvolvedores também podem implementar suas próprias funções e reaproveitá-las em outros programas, desde que as funções tenham sido devidamente testadas e validadas, otimizando o trabalho.

No mundo do desenvolvimento de software, as funções desempenham um papel fundamental na modularização de problemas complexos. Este vídeo explora a importância das funções, tanto as nativas das linguagens de programação quanto as personalizadas pelos desenvolvedores. Descubra como a utilização eficiente de funções pode otimizar o trabalho de desenvolvimento de software e facilitar a implementação de soluções criativas.



Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.



Funções, em linguagens de programação, são pedaços de código, encapsulados, que podem ser chamados em qualquer outro trecho do programa ou do código. Em relação à sua sintaxe, uma função deve ter um nome, uma definição e uma posterior invocação à mesma.

Em uma linguagem, as funções podem ser **nativas** – como a função “**print_r**”, utilizada em alguns exemplos anteriores – ou construídas pelo desenvolvedor – também chamadas de funções definidas pelo usuário.





Dica

Em termos práticos, pense nas funções como um código criado para resolver problemas singulares ou executar tarefas específicas. Além disso, tenha em mente que esses códigos poderão ser usados mais de uma vez ao longo do seu projeto. Logo, em vez de reescrever um mesmo código, faça uso de funções.

O exemplo a seguir descreve a sintaxe de criação de uma função em PHP e a forma de invocá-la.



Conteúdo interativo

esse a versão digital para executar o código.

Como visto no exemplo (aproveite para executá-lo antes de continuar a leitura), a sintaxe de uma função contém os seguintes elementos:

- Palavra reservada “**function**” seguida do nome da função.
- Nome da função seguido de parênteses – “**()**”. Caso receba parâmetros, eles deverão ser declarados dentro dos parênteses. Do contrário, deverão ficar sem conteúdo.
- Instruções da função envoltas em chaves – “**{}**”.

As funções em PHP podem ou não retornar resultados. A primeira delas, “**soma**”, por meio do operador “**return**”, devolve o resultado da soma entre os dois parâmetros recebidos. Repare que a variável “\$num3” recebe justamente esse resultado. Já a função “**imprimir_resultado**” não retorna valores, apenas imprimindo uma frase na tela.



Atenção

Outra particularidade em PHP é que as funções não precisam estar definidas para serem invocadas. Repare que chamamos as duas antes mesmo de codificá-las. Devemos, porém, nos atentar para a quantidade de parâmetros a serem passados. Além disso, sua sintaxe é simples: “nome-da-funcao(parâmetros)” ou “nome-da-funcao()”.

Nomenclaturas de funções e outras boas práticas

A nomeação de funções em PHP segue as mesmas regras para a definição de variáveis, com alguns padrões utilizados e que são considerados como boas práticas. As mesmas dicas cabem, portanto, aqui:

Você pode criar nomes de funções separando nomes compostos por underscore “_” ou como CamelCase, por exemplo. Defina um padrão e siga-o por todo o seu código.

Sobre a sintaxe da função, mais precisamente sobre o posicionamento da chave de abertura, há duas vertentes defendendo que:

1º Vertente

A chave de abertura deve ser inserida logo após o fechamento dos parênteses que guardam os parâmetros da função.



2º Vertente

A chave de abertura deve ser inserida na linha seguinte – essa foi a aplicada nos códigos anteriores.

Por fim, outra boa prática recomendada: **indente seu código** – não só nas instruções inseridas dentro das funções, mas ao longo de todo o programa. Veja o código de exemplo e perceba que as instruções dentro da função não estão coladas no início da linha. Indentar um código ajuda na sua compreensão e no seu entendimento, além de deixar clara a hierarquia existente.

Funções nativas

A linguagem PHP disponibiliza uma série de **funções nativas**, para as mais variadas necessidades. Há funções para a manipulação de **arrays** (como as que vimos nos exemplos anteriores), de **strings**, de arquivos, de acesso de banco de dados, entre tantas outras. A documentação oficial da linguagem ou Manual do PHP é uma fonte extensa e que deve ser sempre consultada.



Saiba mais

A Zend, fabricante norte-americana de software orientado para PHP, é uma das empresas mais conhecidas e relevantes sobre o assunto. Além de um Framework bastante conhecido, que leva o seu nome, ela também é responsável pela certificação profissional PHP mais importante do mercado. Entre todo o material que disponibiliza, há um manual de boas práticas com uma série de convenções relacionadas à produção de código em PHP. Vale a pena a leitura desse material, conforme sugerido na seção Explore +.

Atividade 3

Analise as sentenças seguintes e marque a opção correta.

A

Ao escrever códigos em PHP, só podemos utilizar as funções nativas, tendo em vista que não é possível implementar novas funções.

B

Ao escrever códigos em PHP, só podemos utilizar funções que implementamos, tendo em vista que PHP não tem funções nativas.

C

PHP não tem funções nativas e não permite a implementação de funções.

D

Ao escrever códigos em PHP, é possível utilizar as funções nativas, assim como implementar as próprias funções, otimizando o trabalho de desenvolvimento de software.

E

Em PHP, o uso de funções deve dar lugar ao uso de arrays, que são mais eficientes do ponto de vista computacional, ou seja, são duas abordagens diferentes para se resolver problemas semelhantes.



A alternativa D está correta.

PHP, assim como outras linguagens, tem funções nativas e possibilita a implementação de novas funções, facilitando o reaproveitamento de código e otimizando o trabalho de desenvolvimento de software.

Programando com funções em PHP

As funções são muito utilizadas no desenvolvimento de software, pois possibilitam a divisão dos programas em partes menores, de mais fácil entendimento, implementação e testes. Além disso, favorecem o reaproveitamento de código, otimizando o trabalho. Portanto, o bom entendimento de como implementar e

utilizar as funções, combinado com outros recursos de programação, é fundamental no trabalho do desenvolvedor. PHP tem várias funções nativas, mas permite a criação de novas funções. Considere, ainda, a possibilidade de se utilizar funções associadas a arrays. Por exemplo, suponha que os parâmetros de entrada de uma função sejam elementos de um array. Como fazer isso? É simples. Basta referenciar os elementos dos arrays por meio dos seus índices, de modo que cada elemento se comporte como se fosse uma variável convencional.

Descubra neste vídeo como as funções podem facilitar o desenvolvimento de software, dividindo programas em partes menores e otimizando o trabalho dos desenvolvedores. Aprenda a implementar e utilizar funções em PHP, explorando as funções nativas e a criação de novas funções. Saiba como associar funções a arrays e utilizar elementos de arrays como parâmetros de entrada. Assista ao vídeo e aprimore suas habilidades de programação!



Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.



Roteiro de prática 2

O uso de arrays é muito comum no desenvolvimento de software. Para exemplificar, vamos considerar um array unidimensional, que contém o nome de um aluno, além de suas notas em duas provas. Devemos implementar uma função que calcule a média das duas notas e, depois, que mostre uma mensagem informando o nome do aluno e sua média.

De início, mostraremos o programa completo e, posteriormente, vamos analisá-lo com mais detalhes.

```
plain-text
```



Antes de usarmos alguma função que não é nativa, precisamos implementá-la. O trecho de código compreendido entre as linhas 7 a 9 mostra a implementação da função que calcula a média de dois números passados como parâmetros de entrada dentro dos parênteses, conforme mostrado na imagem seguinte.


```
7 | function calc_media($n1,$n2) {  
8 |     return ($n1 + $n2)/2;  
9 | }
```

Implementação da função que calcula a média de dois números.

Na sequência, é definido o array que contém os dados do aluno, como se vê na próxima imagem.

```
11 | $aluno = array("João", 8.8, 9.4);
```

Definição do array com os dados do aluno.

Depois, é feita a chamada da função que calcula a média das notas do aluno. Observe que os parâmetros de entrada da função são os elementos de índice 1 e 2 do array \$aluno.

```
14 | $media = calc_media($aluno[1], $aluno[2]);
```

Chamada da função que calcula a média das notas do aluno.

Por fim, o nome e a média do aluno são mostrados na tela para o usuário, de acordo com a imagem a seguir. Observe que o nome do aluno foi obtido a partir do índice zero do array \$aluno.

```
17 | echo $aluno[0]." tem média igual a ".$media;
```

Código para mostrar o nome e a média do aluno.

Atividade 4

Suponha que você está participando de um processo seletivo para estágio e, em uma das etapas, você deve analisar o código mostrado a seguir. Depois, marque a opção que descreva a saída do programa, isto é, o que acontece ao executar o código.

php

A

2 4 6 8 10

B

1 3 5 7 9



2 4 6 8



1 3 5 7 9 10



0 2 4 6 8 10



A alternativa A está correta.

Este programa utiliza funções para modularizar o código, tornando-o mais organizado e reutilizável. A função `xpto()` pode ser reutilizada em diferentes contextos para verificar se um número é par. Já a função `abcd()` pode ser usada para exibir números pares em qualquer intervalo especificado.

4. Conclusão

Considerações finais

- Conceitos básicos de PHP.
- Operadores em PHP.
- Estruturas de decisão e de repetição em PHP.
- Vetores e funções.

Podcast

Para encerrar, ouça mais sobre os principais pontos trabalhados no tema e comentários sobre boas práticas de programação em PHP.



Conteúdo interativo

Acesse a versão digital para ouvir o áudio.

Explore +

Para saber mais sobre JavaScript, leia o livro **JavaScript: The Definitive Guide**, de David Flanagan.

Acesse o site **Apache Friends** para aprofundar seus conhecimentos sobre o XAMPP, o mais popular ambiente de desenvolvimento PHP.

Para testar seus códigos PHP, utilize os sites On-line PHP Editor, PHPTester e Write PHP On-line.

Acesse o Manual do PHP e leia os textos:

O que as referências fazem

Escopo de variáveis

Precedência de operadores

Operadores

for

break

continue

Visite o site da comunidade **Mozilla** e pesquise sobre os temas: HTTP; GET; e POST.

Acesse o site **W3Schools** e leia: **The GET method** e **The POST method**.

Leia o manual **Zend Framework coding standard for PHP**, da Zend Framework.

Referências

PHP. **Manual do PHP**: o que é o PHP? Consultado na Internet em: 16 ago. 2020.

