PHP

Professor Claudio Fico
E-mail: cfico@globo.com

Introdução

PHP

O que é PHP?

O PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e links. A diferença de PHP com relação a linguagens semelhantes a Javascript é que o código PHP é executado no servidor, sendo enviado para o cliente apenas html puro.

Desta maneira é possível interagir com bancos de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial.
O que diferencia PHP de um script CGI escrito em C ou Perl é que o código PHP fica embutido no próprio HTML, enquanto no outro caso é necessário que o script CGI gere todo o código HTML, ou leia de um outro arquivo.

PHP

Como surgiu a linguagem PHP?

A linguagem PHP foi concebida durante o outono de 1994 por Rasmus Lerdorf. As primeiras versões não foram disponibilizadas, tendo sido utilizadas em sua home-page apenas para que ele pudesse ter informações sobre as visitas que estavam sendo feitas.

A primeira versão utilizada por outras pessoas foi disponibilizada em 1995, e ficou conhecida como "Personal Home Page Tools" (ferramentas para página pessoal). Era composta por um sistema bastante simples que interpretava algumas macros e alguns utilitários que rodavam "por trás" das home-pages: um livro de visitas, um contador e algumas outras coisas.

PHP

Em meados de 1995 o interpretador foi reescrito, e ganhou o nome de PHP/FI, o "FI" veio de um outro pacote escrito por Rasmus que interpretava dados de formulários HTML (Form Interpreter).
Ele combinou os scripts do pacote Personal Home Page Tools com o FI e adicionou suporte a mSQL, nascendo assim o PHP/FI, que cresceu bastante, e as pessoas passaram a contribuir com o projeto.
Estima-se que em 1996, PHP/FI estava sendo usado por cerca de 15.000 sites pelo mundo, e em meados de 1997 esse número subiu para mais de 50.000. Nessa época houve uma mudança no desenvolvimento do PHP.
Ele deixou de ser um projeto de Rasmus com contribuições de outras pessoas para ter uma equipe de desenvolvimento mais organizada. O interpretador foi reescrito por Zeev Suraski e Andi Gutmans. Esse novo interpretador foi à base para a versão 3.

PHP

O lançamento do PHP4, ocorrido em 22/05/2000, trouxe muitas novidades aos programadores de PHP.

Uma das principais foi o suporte a sessões, bastante útil pra identificar o cliente que solicitou determinada informação.
Além das mudanças referentes a sintaxe e novos recursos de programação, o PHP4 trouxe como novidade um otimizador chamado Zend, que permite a execução muito mais rápida de scripts PHP.

A empresa que produz o Zend promete para este ano o lançamento de um compilador de PHP. Códigos compilados serão executados mais rapidamente, além de proteger o fonte da aplicação. Hoje, já se encontra disponível no mercado a versão 7.x do PHP.

PHP

Características da Linguagem PHP

1 - É uma linguagem de fácil aprendizado;
2 - Tem suporte a um grande número de bancos de dados como: dBase, Interbase, mSQL, mySQL, Oracle, Sybase, PostgreSQL e vários outros.
3 - Tem suporte a outros serviços através de protocolos como IMAP, SNMP, NNTP, POP3 e, logicamente, HTTP;
4 - É multiplataforma, tendo suporte aos sistemas Operacionais mais utilizados no mercado;
5 - Seu código é livre, não é preciso pagar por sua utilização e pode ser alterado pelo usuário na medida da necessidade de cada usuário
6 - Não precisa ser compilado.

Sintaxe

PHP

O código PHP fica embutido no próprio HTML. O interpretador identifica quando um código é PHP pelas seguintes tags:

```
<?php
    código / instruções.....
?>

<script language="php">
    comandos;
</script>
```

PHP

Locais de uso do PHP

```
<?php
   // Sim também podemos ter código PHP antes do DocType.
?>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <title>Titulo</title>
      <?php
         // Sim também podemos ter código PHP no <head>.
      ?>
   </head>
   <body>
      <?php
         // Sim também podemos ter código PHP no <body>.
      ?>
   </body>
</html>
<?php
   // Sim também podemos ter código PHP depois do fechamento da tag <html>.
?>
```

PHP

Separador de instruções

Entre cada instrução em PHP é preciso utilizar o **ponto-e-vírgula (;)**, assim como em C, Perl e outras linguagens mais conhecidas. Na última instrução do bloco de script não é necessário o uso do ponto-evírgula, mas por questões estéticas recomenda-se o uso sempre.

Comentários

Há dois tipos de comentários em código PHP:

Comentários de uma linha:

Marca como comentário até o final da linha ou até o final do bloco de código PHP – o que vier antes. Pode ser delimitado pelo caractere **"#"** ou por duas barras **(//)**. O delimitador "//", normalmente, é o mais utilizado.

Enviando informações ao browser

PHP

Imprimindo código html

Um script php geralmente tem como resultado uma página html, ou algum outro texto. Para gerar esse resultado, deve ser utilizada uma das funções de impressão, **echo e print**.

Sintaxes:

```
echo (argumento1, argumento2, ... );
echo argumento;
ou
print (argumento);
```

PHP

Exemplo:

```
<?php
    echo “teste”; // este teste é similar ao anterior
    echo “teste”; # este teste é similar ao anterior
    // sql “teste”;
?>
```

PHP

Exemplos:

```
<?php
    $a = 10;
    $b = 20;
    $soma = $a + $b; #Imprimindo a soma na tela com o comando echo.
    print ("A soma é: ". $soma); //usando o print para exibir as informações
    echo "<br>"; //quebrando uma linha
    echo ("A soma é: ". $soma); //usando o print para exibir as
    informações
?>
```

PHP

Comentários de mais de uma linha:

Tem como delimitadores os caracteres **"/*" para o início do bloco e "*/" para o final do comentário**. Se o delimitador de final de código PHP (?>) estiver dentro de um comentário, não será reconhecido pelo interpretador.

Exemplo:

```
<?
    echo "teste";
    /* este é um comentário com mais de uma linha.
    echo "teste. Este comando echo é ignorado pelo interpretador do
    PHP por ser um comentário."*/
?>
```

Variáveis

PHP

Nomes de variáveis

Toda variável em PHP tem seu nome composto pelo caracter **$(dólar)** e uma string, que deve iniciar por uma letra ou o caracter "_". O PHP é **case sensitive**, ou seja, as variáveis $vivas e $VIVAS são diferentes.
Por isso é preciso ter muito cuidado ao definir os nomes das variáveis. É bom evitar os nomes em maiúsculas, pois como veremos mais adiante, o PHP já possui algumas variáveis pré-definidas cujos nomes são formados por letras maiúsculas.

Exemplo de variáveis:

1 - $teste = nome correto de variável
2 - teste = nome errado de variável.

PHP

Tipos Suportados

O PHP suporta os seguintes tipos de dados:

1 - Inteiro
2 - Ponto flutuante
3 - String
4 - Array
5 - Objeto

O PHP utiliza checagem de tipos dinâmica, ou seja, uma variável pode conter valores de diferentes tipos em diferentes momentos da execução do script. Por este motivo não é necessário declarar o tipo de uma variável para usá-la. O interpretador PHP decidirá qual o tipo daquela variável, verificando o conteúdo em tempo de execução. Ainda assim, é permitido converter os valores de um tipo para outro desejado, utilizando o ***typecasting*** ou a função ***settype***

PHP

Inteiros (integer ou long)

Uma variável pode conter um valor inteiro com atribuições que sigam as seguintes sintaxes:

1 - $num = 1234; # inteiro positivo na base decimal
2 - $num = -234; # inteiro negativo na base decimal
3 - $num = 0234; # inteiro na base octal-simbolizado pelo 0 equivale a 156 decimal
4 - $num = 0x34; # inteiro na base hexadecimal(simbolizado pelo 0x) – equivale a 52 decimal.

A diferença entre **inteiros simples** e **long** está no número de bytes utilizados para armazenar a variável.
Como a escolha é feita pelo interpretador PHP de maneira transparente para o usuário, podemos afirmar que os tipos são iguais.

PHP

Ponto Flutuante (double ou float)

Uma variável pode ter um valor em ponto flutuante com atribuições que sigam as seguintes sintaxes:

1 - $num = 1.234;

2 - $num = 23e4; # equivale a 230.000

PHP

Strings

Strings podem ser atribuídas de duas maneiras:

a) utilizando aspas simples (') → Desta maneira, o valor da variável será exatamente o texto contido entre as aspas (com exceção de \\ e \' – ver exemplo abaixo)

b) utilizando aspas duplas (") → Desta maneira, qualquer variável ou caractere de escape será expandido antes de ser atribuído.

PHP

Exemplo:

```
<?php
    $texto = "Curso de PHP";
    $teste = "Mauricio";
    $vivas = '---$teste--\n';
    echo "$vivas"; # A saída desse script será "---$teste--\n".
?>

<?php
    $texto = "Curso de PHP";
    $teste = "Mauricio";
    $vivas = "---$teste--\n";
    echo "$vivas"; # A saída desse script será "Mauricio".
?>
```

PHP

Exemplo:

```
<?php
	$titulo = 'Título da minha página';
	$css = '<link rel="stylesheet" type="text/css" href="css/estilos.css" />';
	$conteudo = 'Sejam bem-vindos ao mundo do PHP <br> Nele poderemos ter muitas oportunidades <br><br> <strong>Vamos em
	frente!</strong>';
?>
<!DOCTYPE html>
<html>
	<head>
		<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
		<title><?php echo $titulo; ?></title>
		<?php
			echo $css;
		?>
	</head>
	<body>
		<?php
			echo $conteudo;
		?>
		<p>Eu posso repetir o valor da variável sempre que eu quiser sabia? Veja aqui o nosso titulo aparecendo "<?php echo $titulo; ?>"</p>
		<p>E não necessariamente deveria imprimir <b>$titulo</b> apenas porque eu a declarei primeiro poderia imprimir <b>$css</b> antes
		e depois <b>$titulo</b> ou qualquer uma variável isso vai de acordo com sua necessidade</p>
		<br>
		<p>Agora irei exibir o <b>$conteudo</b> novamente, olhe ela ai:</p>
		<p><?php echo $conteudo; ?></p>
	</body>
</html>
```

PHP

Exercício:

Criar um arquivo com extensão php

1 - Criar antes do DOCTYPE três variáveis e em cada uma das variáveis colocar um determinado conteúdo. Uma das variáveis deverá ser a do título;
2 - Na tag <title> colocar o conteúdo da variável título;
3 - Na tag <body> colocar em cada parágrafo a ser criado por você o conteúdo da variável. Lembrando que como são duas variáveis deverão existir dois parágrafos;
4 - A variável deverá estar dentro do texto a ser criado por você.

PHP

Valores com tipos diferentes em variável no PHP

Obtém o valor de retorno da variável.

```
<?php
    $nome = "Elton Fonseca";
    echo gettype($nome); // string
    echo "<br>";
    $nome = 28;
    echo gettype($nome); // integer
    echo "<br>";
    $nome = 78.500;
    echo gettype($nome); // float
    echo "<br>";
    $nome = true;
    echo gettype($nome); //boolean
?>
```

PHP

Constante

Como você já deve ter imaginado as constantes no PHP guardam valores que nunca serão alterados. Diferente das variáveis que possuem valores que podem ser alterados, sendo assim após definida uma constante ela não pode ser alterada ou removida.
Para definirmos uma constante utilizamos o comando **define()**;

Sintaxe:

```
define( 'NOME_DA_CONSTANTE', 'VALOR DA CONSTANTE' );
```

O nome de uma constante tem a mesma regra de qualquer identificador PHP, ou seja, as mesmas regras de nomes de variáveis exceto pelo fato de constantes **não** iniciarem o nome com **cifrão ($)**.

PHP

Exemplo:

```
<?php
    $titulo = 'Utilizando Constantes';
    $nome = 'Sandro';
    $nascimento = '07/09/1979';
    $sobre_nome = 'Nobrega Tavares';
    define( 'ESTADO', 'Rio de Janeiro' )
?>
<!DOCTYPE html>
<html>
    <head>
            <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
            <title><?php echo $titulo; ?></title>
    </head>
    <body>
            <p><?php echo $nome; ?> <?php echo $sobre_nome; ?>, nascido em <?php echo $nascimento;
            ?>, nasceu no <?php echo ESTADO; ?></p>
    </body>
</html>
```

PHP

Transformação explícita de tipos

A sintaxe do **typecast** de PHP é semelhante ao C: basta escrever o tipo entre parênteses antes do valor.

Os tipos permitidos na **Transformação explícita** são:

1 - (int), (integer) → muda para integer;
2 - (real), (double), (float) → muda para float;
3 - (string) → muda para string;
4 - (array) → muda para array;
5 - (object) → muda para objeto.

PHP

Exemplo:

```
<?php
    $vivas = 15; // $vivas é integer (15)
    $vivas = (double) $vivas; // $vivas é double (15.0)
    $vivas = 3.9; // $vivas é double (3.9)
    $vivas = (int) $vivas; // $vivas é integer (3)

    // o valor decimal é truncado
    print ("O valor de vivas é: $vivas");
?>
```

PHP

Função settype

A função settype converte uma variável para o tipo especificado, que pode ser “integer”, “double”, “string”, “array” ou “object”.

Sintaxe:

settype(nomedavariável, “novo tipo da variávbel”);

O novo tipo da variável deve estar entre aspas.

Exemplo:

```
<?php
    $vivas = 15; // $vivas é integer
    settype($vivas,"double"); // $vivas é double
    $vivas = 15; // $vivas é integer
    settype($vivas,"string"); // $vivas é string
    echo "O valor de vivas é: $vivas";
?>
```

Arrays e Listas

PHP

Arrays

Arrays em PHP podem ser observados como mapeamentos ou como vetores indexados. Mais precisamente, um valor do tipo array é um dicionário onde os índices são as chaves de acesso. O índice terá seu **valor inicial 0**.

Vale ressaltar que os índices podem ser valores de qualquer tipo e não somente inteiros.

Inclusive, se os índices forem todos inteiros, estes não precisam formar um intervalo contínuo como a checagem de tipos em PHP é dinâmica, valores de tipos diferentes podem ser usados como índices de array, assim como os valores mapeados também podem ser de diversos tipos.

PHP

Exemplo:

```
<?php
    $cor[0] = "amarelo";
    $cor[1] = "vermelho";
    $cor[2] = "verde";
    $cor[3] = "azul";
    $cor[4] = "anil";
    echo $cor[4];
?>
```

Equivalentemente, pode-se escrever:

```
<?php
    $cor = array(0 => "amarelo", 1 => "vermelho", 2 => "verde", 3 => "azul",
    4 => "anil");
    echo $cor[2];
?>
```

PHP

Listas

As listas são utilizadas em PHP para realizar atribuições múltiplas. Através de listas é possível atribuir valores que estão num array para variáveis.

Exemplo:

```
<?php
    list($a, $b, $c) = array("a", "b", "c");
    echo "$a<br>";
    echo "$b<br>";
    echo "$c<br>";

    $arr = array(0 => "zero", 1=>"um", 2=>"dois", 3=>"tres");
    list($a, $b, $c, $d) = $arr;
    echo "$a<br>";
    echo "$b<br>";
    echo "$c<br>";
    echo "$d<br>";
?>
```

PHP

Listas

As listas são utilizadas em PHP para realizar atribuições múltiplas. Através de listas é possível atribuir valores que estão num array para variáveis.

Exemplo:

```
list($a, $b, $c) = array(" a ", " b ", " c ");
```

O comando acima atribui valores às três variáveis simultaneamente. É bom notar que só são atribuídos às variáveis da lista os elementos do array que possuem índices inteiros e não negativos. No exemplo acima as três atribuições foram bem sucedidas porque ao inicializar um array sem especificar os índices eles passam a ser inteiros, a partir do zero.

PHP

Exemplo: exibindo a idade de uma pessoa

```
<?php
    $idade = 30;
    $nome = "Clara Machado de Assis"
?>

<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8">
            <title>Teste</title>
    </head>
    <body>
        <strong>Nome: </strong>
        <?php
            echo $nome;
        ?>
        <br>
        <strong>Idade: </strong>
        <?php
            echo $idade;
        ?>
    </body>
</html>
```

Operadores

PHP

Operadores Aritméticos

Só podem ser utilizados quando os operandos são números (integer ou float). Se forem de outro tipo, terão seus valores convertidos antes da realização da operação.

São eles:

Operador	Descrição
+	Adição
-	Subtração
/	Divisão
*	Multiplicação
%	Resto da Divisão

PHP

Exemplo:

```
<?php
    // Declarando os valores das variáveis
    $a = 4;
    $b = 2;
?>

<h2>Adição</h2>
<p>
<?php
    echo $a + $b;
?>
</p>

<h2>Subtração</h2>
<p>
<?php
    echo $a - $b;
?>
</p>
```

PHP

Exemplo:

```
<?php
    // Declarando os valores das variáveis
    $a = 4;
    $b = 2;
?>
<h2>Multiplicação</h2>
<p>
<?php
    echo $a * $b;
?>
</p>

<h2>Divisão</h2>
<p>
<?php
    echo $a / $b;
?>
</p>
```

PHP

Exemplo:

```
<?php
    // Declarando os valores das variáveis
    $a = 4;
    $b = 2;
?>

<h2>Módulo(resto da divisão)</h2>
<p>
<?php
    echo $a % $b;
?>
</p>
```

PHP

Operadores de Strings

Só há um operador exclusivo para strings, que é a **concatenação**

Exemplo:

```
<?php
    $nome = “Clara";
    $sobrenome = "Nascimento";
    echo $nome. " ".$sobrenome;
?>
```

A saída do script acima será: Clara Nascimento

A colocação das aspas entre as variáveis é para proporcionar o espaço entre as palavras.

PHP

Operadores de atribuição

Existe um operador básico de atribuição e diversos derivados. Sempre retornam o valor atribuído. No caso dos operadores derivados de atribuição, a operação é feita entre os dois operandos, sendo atribuído o resultado para o primeiro. A atribuição é sempre por valor, e não por referência.

Operador	Função	Exemplo	Equivalente a
+=	Atribuição e adição	$a += $b	$a = $a + $b
-=	Atribuição e Subtração	$a -= $b	$a = $a - $b
*=	Atribuição e Multiplicação	$a *= $b	$a = $a * $b
/=	Atribuição e Divisão	$a /= $b	$a = $a / $b
%=	Atribuição e Módulo	$a %= $b	$a = $a % $b
.=	Atribuição e concatenação	$a .= $b	$a = $a . $b

PHP

Exemplo com adição:

```
<?php
    $a = 7;
    $a += 2; // $a passa a conter o valor 9
?>
```

Exemplo com subtração:

```
<?php
    $a = 9;
    $a -= 3; // $a passa a conter o valor 6
?>
```

PHP

Operadores bitwise

Comparam dois números bit a bit.

Operador	Descrição
&	“e” lógico
|	“ou” lógico
^	ou exclusivo
~	não (inversão)
<<	shift left
>>	shift right

PHP

Operadores bitwise & (and)

Compara dois valores utilizando suas representações binárias, retornando um novo valor, para formar esse valor de retorno cada bit é comparado, retornando 1(true) quando ambos os bits forem iguais a **1(true)**, caso contrário retorna 0 (false).

Tabela **ASCII** para conferir os valores dos bits referentes aos números inteiros.

O **var_dump()** mostra o conteúdo de uma variável. Se ele for um array é mostrado todo o array estruturado, se for uma variável simples é mostrado o valor dela, se for um objeto então é mostrado todos os campos e assim por diante. O var_dump é muito útil para depuração de código.

PHP

Exemplo:

```
<?php
    $a = 5; // 00000101

    $b = 1; // 00000001

    $c = $a & $b; // 00000101 & 00000001 = 00000001

   var_dump($c); // 00000001 equivale ao valor inteiro 1 na tabela ASC
?>
```

00000101 &
00000001 =
00000001 Resultado

PHP

Exemplo:

```
<?php
    $a = 5; // 00000101
    $b = 3; // 00000011


    $c = $a | $b; // 00000101 | 00000011 = 00000111

    var_dump($c); // 00000111 equivale ao valor inteiro 7 na tabela ASC
?>
```

00000101 |
00000011 =
00000111 Resultado

PHP

Operadores lógicos

Utilizados para inteiros representando valores. São muito utilizados nas estruturas de controle e repetição.

Operador	Descrição
And	“e” lógico
Or	“ou” lógico
Xor	ou exclusivo
!	não (inversão)
&&	“e” lógico
||	“ou” lógico

Existem dois operadores para “e” e para “ou” porque eles têm diferentes posições na ordem de precedência.

PHP

Operadores de comparação

As comparações são feitas entre os valores contidos nas variáveis, e não as referências. Sempre retornam um valor booleano. São muito utilizados nas estruturas de controle e repetição.

Operador	Descrição
==	igual a
!=	diferente de
<	menor que
>	maior que
<=	menor ou igual a
>=	maior ou igual a

PHP

Operadores de Expressão condicional

Existe um operador de seleção que é **ternário**. Funciona assim:
(expressao1)?(expressao2):(expressao3)

O interpretador PHP avalia a primeira expressão. Se ela for verdadeira, a expressão retorna o valor de expressão2. Senão, retorna o valor de expressão3.

Exemplo:

```
<?php
    $a = 4;
    $b = 2;
    echo $a/$b == 2 ? "O resultado da divisão é 2" : "O resultado da
    divisão não é 2";
?>
```

PHP

Operadores de incremento e decremento

++ incremento; e
-- decremento.

Podem ser utilizados de duas formas: antes ou depois da variável. Quando utilizado antes, retorna o valor da variável antes de incrementá-la ou decrementá-la. Quando utilizado depois, retorna o valor da variável já incrementado ou decrementado.