Guia itexto de PHP

Henrique Lobo Weissmann <kico@itexto.com.br>, (entre seu nome e e-mail aqui)

Version 1.0, 9/4/2017

Índice

Introdução

Com o passar do tempo começamos a notar a necessidade de nos aprofundarmos na linguagem PHP e em todo o ecossistema que a envolve. Por esta razão resolvemos escrever este guia: um esforço coletivo no qual iremos buscar a criação de um material de referência que inicialmente visa atender nossas demandas de conhecimento internas para, posteriormente, quem sabe, atender o público geral também.

Este é apenas um esforço inicial, sendo assim não o leve tão a sério, ok?

A linguagem de programação PHP

Sintaxe

Autor: Henrique Lobo Weissmann

Tags PHP (ou como o PHP se detecta)

A linguagem PHP foi criada para que fosse possível adicionar dinamismo às páginas web. Para tal tirou-se proveito da própria estrutura do HTML para que fosse possível mesclar código das duas linguagens (PHP e HTML) em um único arquivo de uma forma que fosse realmente produtiva.

Apesar de soar estranho nos dias de hoje, na época foi algo revolucionário (mesclar código HTML com o da linguagem), pois tornava a escrita de páginas web dinâmicas algo muito mais simples que o existente até então.

Como era escrever páginas dinâmicas antes do PHP

Uma das primeiras formas (a primeira) de se criar páginas web dinâmicas foi o CGI (Common Gateway Interface). Essencialmente, sempre que uma requisição era feita a um servidor, este buscava em seu sistema de arquivos qual programa deveria ser executado e a saída deste era retornada para o browser do navegador.

Não era algo bonito de se fazer. Para se escrever uma página simples em Perl, por exemplo, você precisava digitar algo similar ao código a seguir:

```
#!/usr/bin/perl
print "<html>";
print "  <head>";
print "  <title>Já começou a achar chato?</title>";
print "  <head>";
print "  <body>";
print "   <h1>Uma página escrita em Perl usando CGI</h1>";
print "  </body>";
print "</html>";
```

O programa escrito em CGI podia ser em qualquer linguagem, tais como C, por exemplo. Entretanto a primeira linguagem que se tornou bastante popular para a escrita deste tipo aplicações web foi o Perl.

Dado que PHP é uma linguagem interpretada, o interpretador irá reconhecer código PHP apenas se este estiver contido entre as tags `<?php` e `?>`.

Sendo assim, um olá mundo em PHP pode ser similar ao exposto a seguir:

```
<?php
echo "Oi";
?>
```

Se o arquivo contiver apenas código PHP, recomenda-se que a tag `<?php` não seja fechada, dado que isto evita a ocorrência de caracteres de espaço ou novas linhas serem incluídas ao final do arquivo ao final da tag. Esta é inclusive uma boa prática de programação, tal como referenciado na própria documentação oficial da linguagem [doc-oficial-tags].

Mesclando código PHP e HTML

Na época de lançamento do PHP um dos seus recursos mais falados era a possibilidade de mesclarmos código PHP com HTML. Hoje sabemos (e na época também, mas ignorávamos) que esta não é uma boa prática, dado que o ideal é termos a camada de visualização separada da de negócio.

Dado que o código PHP é executado sempre que a sua tag `<?php` é encontrada pelo interpretador, o código a seguir é perfeitamente válido:

```
<html>
  <head>
    <title>Quanto é 1 + 7?</title>
  </head>
  <body>
    <p>Quanto é 1 + 7?</p>
    <p>Hora: 1 + 7 é <?php echo (1 + 7); ?></p>
  </body>
</html>
```

Cuja saída, no navegador, será a página HTML a seguir:

```
<html>
  <head>
    <title>Quanto é 1 + 7?</title>
  </head>
  <body>
    <p>Quanto é 1 + 7?</p>
    <p>Hora: 1 + 7 é 8</p>
  </body>
</html>
```

É importante saber desta possibilidade: apesar de não ser recomendada, é muito usada na escrita de temas e plugins para o Wordpress, por exemplo.

Variações das tags PHP

É importante salientar que há algumas variações das tags PHP, caso você precise lidar com código legado. Uma delas, muito pouco usada mas sempre presente é exposta a seguir, que foi desabilitada

a partir da versão 7.0 da linguagem:

```
<script language="php">
echo "Código PHP entra aqui";
</script>
```

Há situações nas quais, em código que mescla HTML e PHP, você queira apenas gerar a saída de uma expressão. Para tal, basta usar a forma `<?=` e `?>`. Sendo assim, o exemplo da página acima ficaria tal como exposto a seguir:

```
<html>
  <head>
    <title>Quanto é 1 + 7?</title>
  </head>
  <body>
    <p>Quanto é 1 + 7?</p>
    <p>Hora: 1 + 7 é <?= 1 + 7 ?></p>
  </body>
</html>
```

Importante salientar que esta forma só é habilitada para a versão 5.4.0 ou posterior do PHP.

Que tags devo usar então?

Use `<?=` e `?>` quando quiser expor o resultado de uma expressão e sua versão do PHP for superior ou igual à 5.4.0.

Use `<?php` e `?>` como padrão para tudo, visto que é aceito por todas as versões da linguagem.

Separando instruções

No PHP, assim como em Java e C/C++, as instruções quase sempre são separadas pelo caractere `;`.

Por que sempre? Por que no caso da última expressão do código PHP dentro das tags (chamadas também de blocos) o caractere separador de instruções é opcional.

Exemplos:

```
<?php
  echo "Isto pode";
  echo "Isto também pode"
?>

<?php echo "Isto pode" ?>

<?php
   echo "Isto não pode"
   echo "Por que tem outra linha na sequência";
?>
```

Comentários

Você vai querer comentar seu código. Sendo assim, é importante saber como PHP lida com este recurso da linguagem.

Você tem comentários no formato C padrão e também no padrão Perl. Vamos primeiro ver exemplos de comentários no formato C/Java:

```
echo "Eu vou ser executado"; // eu sou um comentário de uma linha

// um comentário de uma linha

/*
  Um comentário com mais de uma linha
  que voce pode ver
  aqui.
*/
echo "Eu vou ser executado também."
```

Talvez você não conheça o formato Perl de comentários. É bem simples: basicament é a mesma coisa que o comentário de uma única linha do C, só que usamos o caractere #, tal como exposto no exemplo a seguir:

```
echo "Vou rodar!"; # errado: você vai executar, quem roda é pneu!

# um comentário de uma linha no formato Perl.
```

Sistema de tipos

Toda linguagem possui seu próprio sistema de tipos. Segue uma descrição sucinta dos principais fornecidos pelo PHP.

Tal como Perl, PHP não possui tipagem estática. O tipo de uma variável pode mudar durante a execução do programa. Entretanto, isto não quer dizer que variáveis não possuam tipos ou regras

relacionadas a estes.

Há quatro tipos escalares ("primitivos", no jargão do Java): `boolean`, `integer`, `float` (ou `double`, dependendo de onde você ler na documentação oficial da linguagem e literatura relacionada) e `string`.

Há quatro tipos compostos: `array`, `object`, `callable` e `iterable`.

E dois tipos especiais: `resource` e `NULL`.

Tipos escalares

Booleanos

O tipo mais simples do PHP: há dois valores possíveis: `TRUE` e `FALSE`, sendo que seus valores são `case insensitive`, tal como exposto no exemplo a seguir:

```
$booleano = true; // verdadeiro
$booleano2 = True; // Verdadeiro
#booleano3 = TRUE; // VERDADEIRO
$booleanof = false; // falso
$booleanof = faLse; // faLso
# creio que você já pegou a ideia
```

Na conversão automatica de tipos do PHP, as seguintes condições serão interpretadas como falso:

- o `FALSE` própriamente dito.
- o valor inteiro zero.
- o valor float 0.0.
- uma string vazia.
- um array vazio.
- o tipo especial NULL

Qualquer outro valor será considerado verdadeiro. Uma nota interessante: `-1` é considerado verdadeiro. Apenas `0` é considerado falso.

Inteiros

O valor inteiro pode ser representado nas bases decimal, hexadecimal, octal e binária. Pode ser positivo ou negativo (quando precedido do caractere `-`).

Não há muito o que ser dito sobre a sintaxe na declaração destes tipos, mas é importante saber como representar valores inteiros nas bases citadas acima:

```
$inteiro_positivo_decimal = 1234;
$inteiro_negativo_decimal = -1234;
$inteiro_positivo_octal = 01234;
$inteiro_negativo_octal = -01234;
$inteiro_positivo_hexadecimal = 0x1B;
$inteiro_negativo_hexadecimal = -0x1B;
$inteiro_positivo_binario = 0b100;
$inteiro_negativo_binario = -0b100;
```

Nota importante sobre limites

Os tamanhos máximos e mínimos para valores inteiros no PHP depende da plataforma no qual este é executado. Sendo assim, é importante conhecer algumas constantes importantes na linguagem:

- **PHP_INT_SIZE** - retorna o tamanho em bits do tipo inteiro.
- **PHP_INT_MAX** - o valor máximo de um número inteiro na plataforma de execução.
- **PHP_INT_MIN** - o valor mínimo de um número inteiro na plataforma de execução.

> **Nota importane sobre overflow**
>
> Se ocorrer um overflow, o valor da expressão retornado será do tipo **float**. Logo, se topar com uma expressão envolvendo inteiros que retorne um valor do tipo **float**, já sabe: ocorreu um overflow na sua expressão.

Ponto flutuante (float, double)

Podem ser representados sob a forma normal, usando o caractere `.` para separar as casas decimais ou no formato científico, tal como exposto nos exemplos a seguir:

```
$float = 1.234;
$float = 1.23e4;
$float = 7E-10;
```

Importante mencionar que o PHP adota o padrão IEEE-754 para a representação de números de ponto flutuante. Mais detalhes sobre o padrão em [ieee-754]. Sendo assim, é fundamental levar em consideração a imprecisão definida pelo formato na realização de operações com ponto flutuante.

String

Tal como na esmagadora maioria das linguagens de programação, são usadas para representar texto. Consitem em uma sequência de caracteres que são agrupados usando-se os caracteres `"` ou `'`.

Fundamental mencionar que o PHP não oferece suporte nativo a caracteres UNICODE, dado que considera um caractere como sendo um byte.

Caso venha a usar o PHP para processar grandes quantidades textuais, é também necessário saber

que o tamanho máximo de uma string até a versão 7.0 da linguagem era de 2 Gigabytes (a partir da 7.0 não há mais esta limitação).

Sintaxe

Àspas simples (')

A forma mais simples de se representar uma string em PHP, tal como pode ser visto no exemplo a seguir:

```
$str = 'Sou uma string';
```

Para usarmos o caractere ' no interior de uma string neste formato, basta usar o caractere de escape \, tal como no exemplo a seguir:

```
$str = 'Sou uma \'string\'.'; # Sou uma 'string'
```

Também é possível ter uma string com mais de uma linha neste formato:

```
$linhas = 'Sou uma string
com mais de uma
linha';
```

Se quiser usar o caractere \, basta precedê-lo por \, tal como em

```
$linhas = 'Lembra do C:\\?'; # Lembra do C:\?
```

Áspas duplas

Usamos àspas duplas quando queremos realizar a interpolação de variáveis. Segue um exemplo prático:

```
$valor1 = 1;
$valor2 = 7;
$texto = "$valor1 + $valor2 = "
```

Heredoc

Uma forma muito prática de se usar strings, especialmente quando queremos criar um template com elas.

Sua sintaxe parece estranha em um primeiro momento. O valor é precedido de \<\<\< e um identificador qualquer (qualquer texto que não contenha espaços). Na sequência, digite o texto que quiser (é possível interpolar variáveis) e termine com uma linha que contenha apenas o identificador, seguido do caractere ;.

Complexo? Vamos a um exemplo então usando o identificador `ITEXTO`.

```
$texto = <<<ITEXTO
Bom, este é um texto
que contém mais de uma linha.
E que serve para ilustrar o uso do HEREDOC.
ITEXTO;
```

Nota importante: a última linha no formato HEREDOC não deve possuir qualquer caractere que não seja o identificador seguido do caractere `;`. Sendo assim, **em hipótese alguma** idente este trecho do seu código fonte.

Interpolação de variáveis

Tanto no formato de áspas duplas quanto no HEREDOC o PHP permite a interpolação de variáveis.

O exemplo abaixo nos mostra como realizar a interpolação de variáveis.

```
$numero = 8;
$texto = "Sou o melhor número. Sou o $numero";
// Sou o melhor número, Sou o 8
```

Concatenação de strings

Concatenamos strings usando o caractere `.`. Veja o exemplo a seguir:

```
$saudacao = "Bom dia ";
$nome = "Jetson";
echo $saudacao . $nome;
// Bom dia Jetson
```

Acessando caracteres específicos em uma string

Logo no início mencionamos que uma string é uma sequência de caracteres. Sendo assim, nada mais natural que acessar um caractere a partir de sua posição.

Pense na string como se fosse um array, que começa na posição 0 e irá seguir até a posição (n - 1), aonde `n` representa o número de caracteres em uma string.

Acessamos um caractere portanto a partir do seu índice com a sintaxe `\[ \]`, exposta no exemplo a seguir:

```
$texto = "01234";
echo $texto[0]; # 0
echo $texto[1]; # 1
echo $texto[4]; # quinta posição, caractere 4
```

Como saber o tamanho de uma string? Use a função `strlen`, tal como no exemplo a seguir.

```
$texto = "Oi Kico";
$tamanho = strlen($texto); # $tamanho = 7
```

Variáveis e estruturas de dados essenciais

Aqui entra o texto sobre como declarar variáveis em PHP, qual o tipo de tipagem, etc.

Entra também a descrição de arrays em php, que é um conceito fundamental.

Orientação a objetos

Como é a orientação a objetos em PHP

Bibliografia

- [doc-oficial] - Documentação Oficial da Linguagem PHP - http://php.net/docs.php
- [doc-oficial-tags] - PHP Tags - Documentação oficial do PHP - http://php.net/manual/en/language.basic-syntax.phptags.php

[ieee-754] - Padrão IEEE 754 - Wikipedia - https://pt.wikipedia.org/wiki/IEEE_754