VOLTAR

Manipulação de Arrays e objetos

Neste tópico apresentaremos as principais funções para manipulação de arrays e os tipos de objetos possíveis no script PHP.

NESTE TÓPICO

Marcar tópico

Manipulação de *arrays*

Essas funções internas do PHP permitem a interação e manipulação de *arrays* de várias formas. *Arrays* são essenciais para armazenar, gerenciar, operar sobre um conjunto de variáveis.

Arrays, objetos e operadores, *arrays* simples (vetores) ou multidimensionais (matrizes) são estruturas de dados que armazenam uma coleção de elementos de tal forma que cada um deles possa ser identificado por, pelo menos, um índice ou uma chave (CONVERSE, T. & PARK, J., 2005).

Apenas para relembrar, uma das maneiras mais comuns para criar um *array* em PHP é utilizando o construtor *array*(), veja um exemplo:

```
1.  <?php
2.      $nome_do_array=array();
3.  ?>
```

No exemplo acima, a variável $nome_do_array já é um *array*! Simples assim.

Se quisermos, também podemos inicializar nosso *array* contendo alguns valores. Por exemplo, vamos inicializar nosso *array* com seis valores.

```
<?php
      $bandas=array("Pearl Jam","Metallica","Nirvana","New Order", "Angra","Faith No More");
?>
```

Mas e se quiséssemos manipular nossos *arrays*? Como colocá-los em ordem alfabética, como faríamos? Abaixo, temos algumas das principais funções para manipulação de *arrays*:

array_unique(): remove valores duplicados de um *array*.

Exemplo:

```
<?php
      $bandas=array ("Pearl Jam","Metallica","Pearl Jam","Faith No More","Nirvana","New Order","Angra","New Order","Faith No More");
      $result=array_unique($bandas);
      foreach($resul as $x=>$x_value){
           echo "Banda[" . $x . "]: " .  $x_value;
           echo "<br>";
      }
?>
```

```
<?php
      $bandas=array ("Pearl Jam","Metallica","Pearl Jam","Faith No More","Nirvana","New Order","Angra","New Order","Faith No More")
      $result=array_unique($bandas);
      foreach($resul as $x=>$x_value){
           echo "Banda[" . $x . "]: " .  $x_value;
           echo "<br>";
      }
?>
```

Na tela do navegador, teremos o seguinte resultado:

Figura 1: Removendo valores duplicados num array.

asort(): ordena um *array* associativo em ordem crescente, de acordo com o valor ou conteúdo da chave.

Exemplo:

```
1.  <?php
2.        $bandas=array("Pearl Jam","Metallica","Nirvana","New Order", "Angra","Faith No
    More");
3.       asort($bandas);
4.       foreach($bandas as $x=>$x_value){
5.            echo "Banda[" . $x . "]: " .  $x_value . "<br>";
6.       }
7.  ?>
```

Na tela do navegador, teremos o seguinte resultado:

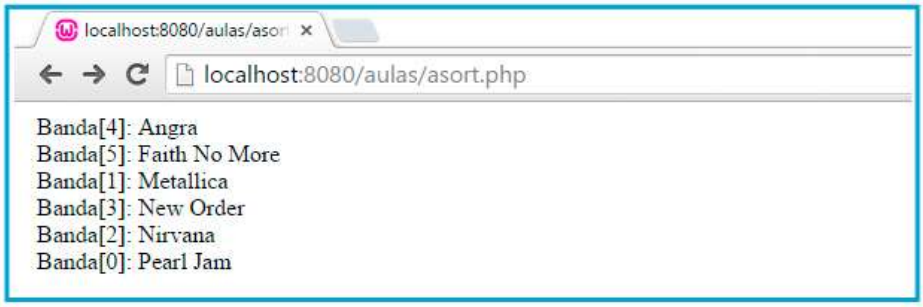

Figura 2: Classificando um array em ordem crescente, de acordo com o conteúdo do índice.

arsort(): ordena um *array* associativo em ordem decrescente, de acordo com o valor ou conteúdo da chave.

Exemplo:

```
1.  <?php
2.        $bandas=array("Pearl Jam","Metallica","Nirvana","New Order", "Angra","Faith No
    More");
3.       arsort($bandas);
4.       foreach($bandas as $x=>$x_value){
5.            echo "Banda[" . $x . "]: " .  $x_value . "<br>";
6.       }
7.  ?>
```

Na tela do navegador, teremos o seguinte resultado:

Figura 3: Classificando um array em ordem decrescente, de acordo com o conteúdo do índice.

count(): retorna o número de elementos dentro de um *array*

Exemplo:

```
<?php
      $bandas=array("Pearl Jam","Metallica","Nirvana","New Order", "Angra","Faith No More");
     $i = count($bandas);
     echo "Foram encontradas $i bandas no array.";
     echo "<br>";
     echo "e são elas:<br>";
     foreach($bandas as $x=>$x_value){
          echo "Banda[" . $x . "]: " .  $x_value;
          echo "<br>";
     }
?>
```

Na tela do navegador, teremos o seguinte resultado:

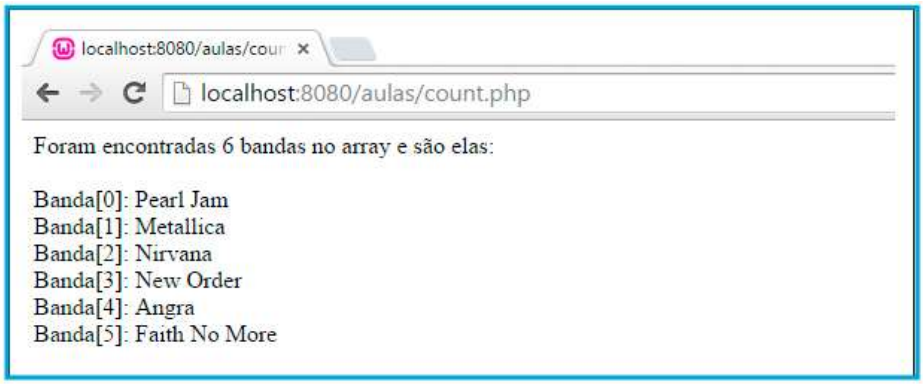

Figura 4: A função count() faz a contagem da quantidade de elementos de um array.

ksort(): ordena um *array* associativo em ordem crescente, de acordo com o valor da chave ou do índice.

Exemplo:

```
<?php
      $bandas=array("Pearl Jam","Metallica","Nirvana","New Order", "Angra","Faith No More");
     ksort($bandas);
     foreach($bandas as $x=>$x_value){
          echo "Banda[" . $x . "]: " .  $x_value . "<br>";
     }
?>
```

Na tela do navegador, teremos o seguinte resultado:

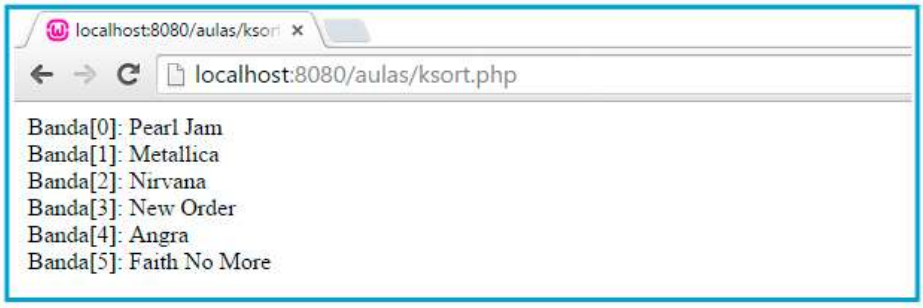

Figura 5: Classificando um array em ordem crescente, de acordo com o índice.

krsort(): ordena um *array* associativo em ordem decrescente, de acordo com o valor da chave ou do índice.

Exemplo:

```
1.  <?php
2.        $bandas=array("Pearl Jam","Metallica","Nirvana","New Order", "Angra","Faith No
    More");
3.       krsort($bandas);
4.       foreach($bandas as $x=>$x_value){
5.            echo "Banda[" . $x . "]: " .  $x_value . "<br>";
6.       }
7.  ?>
```

Na tela do navegador, teremos o seguinte resultado:

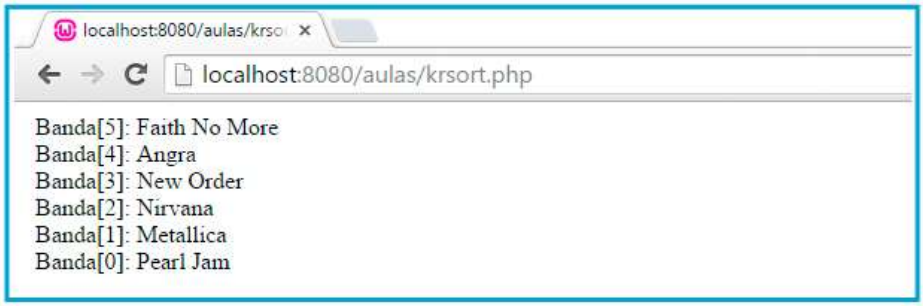

Figura 6: Classificando um array em ordem decrescente, de acordo com o índice.

Abaixo temos uma completa de funções para manipulação de *arrays*, sintaxes e mais informações sobre elas, podem ser encontradas no seguinte endereço: http://www.w3schools.com/php/php_ref_array.asp

FUNÇÃO	DESCRIÇÃO
array_change_key_case( )	Modifica todas as chaves em um array

FUNÇÃO	DESCRIÇÃO
array_chunk( )	Divide um array em pedaços
array_column( )	Devolver os valores de uma única coluna na matriz de entrada
array_combine( )	Cria um array usando um array para chaves e outro para valores
array_count_values( )	Conta as frequências de cada valor de um array
array_diff_assoc( )	Computa a diferença entre arrays com checagem adicional de índice
array_diff_key( )	Registra a diferença entre arrays usando chaves para comparação
array_diff_uassoc( )	Computa a diferença entre arrays com checagem adicional de índice que é feita por uma função de callback fornecida pelo usuário
array_diff_ukey( )	Computa a diferença de arrays usando uma função callback na comparação de chaves
array_diff( )	Analisa as diferenças entre arrays
array_fill_keys( )	Preenche um array com valores, especificando chaves
array_fill( )	Preenche um array com valores
array_filter( )	Filtra os elementos da array usando uma função de callback
array_flip( )	Inverte as relações entre chaves e valores
array_intersect_assoc( )	Computa a interseção de arrays com uma adicional verificação de índice
array_intersect_key( )	Computa a interseção de array comparando pelas chaves
array_intersect_uassoc( )	Computa a interseção de arrays com checagem de índice adicional, compara índices por uma função de callback
array_intersect_ukey( )	Computa a interseção de arrays usando uma função de callback nas chaves para comparação
array_intersect( )	Calcula a interseção entre arrays
array_key_exists( )	Checa se uma chave ou índice existe em um array
array_keys( )	Retorna todas as chaves de um array

FUNÇÃO	DESCRIÇÃO
array_map( )	Aplica uma função em todos os elementos dos arrays dados
array_merge_recursive( )	Une dois ou mais arrays recursivamente
array_merge( )	Une um ou mais arrays
array_multisort( )	Ordena múltiplos arrays ou arrays multidimensionais
array_pad( )	Expande um array para um certo comprimento utilizando um determinado valor
array_pop( )	Retira um elemento do final do array
array_product( )	Calcula o produto dos valores de um array
array_push( )	Adiciona um ou mais elementos no final de um array
array_rand( )	Retorna um ou mais elementos aleatórios de um array
array_reduce( )	Reduz um array para um único valor através de um processo iterativo utilizando uma função
array_replace_recursive( )	Substitui os elementos de matrizes que passaram para a primeira matriz de forma recursiva
array_replace( )	Substitui os elementos de matrizes que passaram para a primeira matriz
array_reverse( )	Retorna um array com os elementos na ordem inversa
array_search( )	Procura por um valor em um array e retorna sua chave correspondente caso seja encontrado
array_shift( )	Retira o primeiro elemento de um array
array_slice( )	Extrai uma parcela de um array
array_splice( )	Remove uma parcela do array e substitui com outros elementos
array_sum( )	Calcula a soma dos elementos de um array
array_udiff_assoc( )	Computa a diferença entre arrays com checagem adicional de índice, compara dados por uma função de callback

FUNÇÃO	DESCRIÇÃO
array_udiff_uassoc( )	Computa a diferença entre arrays com checagem adicional de índice, compara dados e índices por uma função de callback
array_udiff( )	Computa a diferença de arrays usando uma função de callback para comparação dos dados
array_uintersect_assoc( )	Computa a interseção de arrays com checagem adicional de índice, compara os dados utilizando uma função de callback
array_uintersect_uassoc( )	Computa a interseção de arrays com checagem adicional de índice, compara os dados e os índices utilizando funções de callback
array_uintersect( )	Computa a interseção de array, comparando dados com uma função callback
array_unique( )	Remove os valores duplicados de um array
array_unshift( )	Adiciona um ou mais elementos no início de um array
array_values( )	Retorna todos os valores de um array
array_walk_recursive	Aplica uma função do usuário recursivamente para cada membro de um array
array_walk( )	Aplica uma determinada função em cada elemento de um array
array( )	Cria um array
arsort( )	Ordena um array em ordem decrescente mantendo a associação entre índices e valores
asort( )	Ordena um array mantendo a associação entre índices e valores
compact( )	Cria um array contendo variáveis e seus valores
count( )	Conta o número de elementos de uma variável, ou propriedades de um objeto
current( )	Retorna o elemento corrente em um array
each( )	Retorna a chave/valor corrente de um array e avança o seu cursor
end( )	Faz o ponteiro interno de um array apontar para o seu último elemento
extract( )	Importa variáveis para a tabela de símbolos a partir de um array

FUNÇÃO	DESCRIÇÃO
in_array( )	Checa se um valor existe em um array
key_exists( )	Sinônimo de array_key_exists
key( )	Retorna uma chave de um array
krsort( )	Ordena um array pelas chaves em ordem decrescente
ksort( )	Ordena um array pelas chaves
list	Cria variáveis como se fossem arrays
natcasesort( )	Ordena um array utilizando o algoritmo da "ordem natural" sem diferenciar maiúsculas e minúsculas
natsort( )	Ordena um array utilizando o algoritmo da "ordem natural"
next( )	Avança o ponteiro interno de um array
pos( )	Sinônimo de current
prev( )	Retrocede o ponteiro interno de um array
range	Cria um array contendo uma faixa de elementos
reset( )	Faz o ponteiro interno de um array apontar para o seu primeiro elemento
rsort( )	Ordena um array em ordem decrescente
shuffle( )	Mistura os elementos de um array
sizeof( )	Sinônimo de count
sort( )	Ordena um array
uasort( )	Ordena um array utilizando uma função de comparação definida pelo usuário e mantendo as associações entre chaves e valores
uksort( )	Ordena um array pelas chaves utilizando uma função de comparação definida pelo usuário.
usort( )	Ordena um array pelos valores utilizando uma função de comparação definida pelo usuário

PHP Datatypes

O tipo de uma variável geralmente não é definido pelo programador: isto é decidido em tempo de execução pelo PHP, dependendo do contexto na qual a variável é usada.

O PHP suporta os oito tipos primitivos de dados:

- São quatro tipos escalares:
  - *Boolean*: Este é o tipo mais simples. Um *booleano* expressa um valor verdadeiro ou falso, TRUE ou FALSE, ou ainda 0 ou 1 (REIS, 2015).

```
<?php
    $_var=true;
?>
```

  - *Integer*: Um inteiro é um número do conjunto Z = {..., -2, -1, 0, 1, 2, ...}. Inteiros podem ser especificados em notação decimal (base 10), hexadecimal (base 16) ou octal (base 8), opcionalmente precedido de sinal (- ou +) (REIS, 2015).

```
<?php
    $a = 1234; //número decimal
    $a = -123; // um número negativo
    $a = 0123; // número octal (equivalente a 83 em decimal)
    $a = 0x1A; // número hexadecimal (equivalente a 26 em decimal)
?>
```

O tamanho de um inteiro é dependente de plataforma, sendo um número aproximado a 2 bilhões o valor mais comum (número de 32 bits com sinal). O PHP não suporta inteiros sem sinal.

  - *Float* ou *double* (também conhecidos como "ponto flutuante" ou "números reais") podem ser especificados utilizando qualquer uma das seguintes sintaxes:

```
<?php
    $a = 1.234;
    $b = 1.2e3;
    $c = 7E-10;
?>
```

```php
<?php
      $_var=True;
      $a = 1.234;
      $b = 1.2e3;
      $c = 7E-10;
?>
```

A precisão de um número de ponto flutuante é dependente de plataforma, sendo o máximo de ~1.8e308 com uma precisão de 14 dígitos decimais um valor comum (número de 64 bits no formato IEEE) (CONVERSE, T. & PARK, J, 2005)..

- *String*: Uma *string* é uma série de caracteres.

```php
<?php
      $_var="Helo world!!";
?>
```

Não é problema para o PHP uma *string* ser bastante longa. Não há imposição de limite no tamanho de uma *string*; o único limite é o de memória disponível do computador no qual o PHP está sendo executado (REIS, 2015).

- Dois tipos compostos:
  - *Array*: Um *array* no PHP é atualmente um mapa ordenado. Um mapa é um tipo que relaciona valores para chaves.

```php
<?php
      $arr = array("foo" => "bar", 12 => true);
      echo $arr["foo"];
      echo $arr[12];
?>
```

Não entraremos em muitos detalhes sobre *arrays*, pois dedicaremos um tópico específico para eles.

- *Object*: Para criar um novo objeto, use a instrução *new* para instanciar uma classe:

```php
<?php
      class teste
      function hello(){
           echo "Hello World!!";
      }
      $var = new teste;
      $var-> hello ();
?>
```

Mais informações sobre o tipo *objects* pode ser encontrado em: http://php.net/manual/pt_BR/language.oop5.php

- E finalmente dois tipos especiais:
    - *Resource*: Um recurso é uma variável especial, que mantém uma referência a um recurso externo. Recursos são criados e usados por funções especiais (REIS, 2015).

Mais informações sobre o tipo *resources*, pode ser encontrado em: http://php.net/manual/pt_BR/resource.php

    - *NULL*: O valor especial *NULL* representa que a variável não tem valor.

```
1.  <?php
2.        $var = NULL;
3.  ?>
```

Vale ressaltar que a palavra-chave NULL é *case-insensitive* ou seja podemos escreve-la em maiúsculas ou minúsculas (REIS, 2015).

Espero que o conteúdo tenha sido proveitoso. Estudem e até o próximo tópico!

Quiz

Exercício Final

Manipulação de Arrays e objetos

INICIAR ❯

Referências

CONVERSE, T. & PARK, J. **PHP 5 - A Bíblia**. 2. Ed. Rio de Janeiro: Campus, 2005.

DALL'OGLIO, P. **Programando com orientação a objetos (Inclui *Design Patterns*)**. São Paulo: Novatec, 2009.

MARTIN, J. **Princípios de análise e projeto baseado em objetos**. Rio de Janeiro: Editora Campus, 1994.

MATHEUS, D. Variáveis e constantes no PHP. Disponível em <http://www.diogomatheus.com.br /blog/php/variaveis-e-constantes-no-php/>, acessado em 19/09/2015, às 12h15min. Publicado em 26/09/2012.

MUTO, C A. **PHP e MySQL: Guia Introdutório**. 3. Ed. Rio de Janeiro: Brasport, 2006.

NIEDERAUER, J. **Desenvolvendo Websites com PHP**. 2. Ed. São Paulo: Novatec, 2004.

PHP. **PHP: Introdução. Manual PHP.** Disponível em < http://php.net/manual/pt_BR/language. types.intro.php>. Acessado em 04/10/2015, às 10h.

PHP. **PHP: Funções para array - Manual.** Disponível em <http://php.net/manual/pt_BR/ref. array.php>. Acessado em 04/10/2015, às 16h55min.

PHP. **PHP: Introdução. Manual PHP.** Disponível em <http://php.net/manual/pt_BR/language. variables.basics.php>. Acessado em 04/10/2015, às 14h20min.

REIS, Fábio. **Declarando e usando Variáveis em PHP.** Disponível em <http://www. bosontreinamentos.com.br/php-programming/curso-de-php-declaracao-e-atribuicao-de-variaveis/>, acessado em: 16/09/2015, às 11h30min. Publicado em 23/06/2015.

SOARES, L.; AUGUSTO, B. **Aprendendo a Linguagem PHP**. 1. Ed. Rio de Janeiro: Ciência Moderna, 2007.

SOARES, W. **PHP 5 - Conceitos, Programação e Integração com Banco de Dados**. 4. Ed. São Paulo: Érica, 2004.

TANSLEY, D. **Como criar *Web Pages* rápidas e eficientes usando PHP e MySQL**. 1. Ed. Rio de Janeiro: Ciência Moderna, 2002.

Avalie este tópico

ANTERIOR
Estruturas de Decisão e Looping

Índice

PRÓXIMO
Variáveis pré-definidas

Ajuda?
(https://ava.uninove.br/seu/AVA/topico/topico.php?idCurso=)

Biblioteca
(https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)
Portal Uninove
(http://www.uninove.br)
Mapa do Site