Aula 2º

# Estruturas Avançadas I

Olá,

Vamos continuar os nossos estudos da matéria Desenvolvimento Voltado a Web II? Na aula anterior, foi apresentado o funcionamento do PHP, que funciona no lado do servidor, interpretando e construindo um documento para ser enviado para o lado do cliente. Além disso, foram exemplificadas a sintaxe básica, os operadores e estruturas de controle.

Na aula de hoje, faremos um estudo sobre como manipular estruturas mais avançadas, como os arrays e os arquivos.

Leia atentamente esta aula e, se tiver alguma dúvida, use os recursos que estão na sua área do aluno.

Bons estudos!

## Objetivos de aprendizagem

Ao término desta aula, vocês serão capazes de:

- criar, armazenar e recuperar arrays;
- manipular arquivos.

## Seções de estudo

1 – Arrays
2 – Arquivos

# 1- Arrays

Na aula anterior, foi feita uma introdução aos arrays para poder explicar o funcionamento do foreach. Nesta aula, veremos mais exemplos de utilização dos arrays.

Um array é uma variável especial. Na documentação PHP (2019) ela é chamada de um mapa ordenado, que relaciona valores a chaves. Essa estrutura é otimizada para ser utilizada como array, lista (vetor), hashtable, dicionário, coleção, pilha, fila e mais. Além dessas, ainda existe a possibilidade de ser utilizado como árvores e arrays multidimensionais, quando os valores do array forem outros arrays.

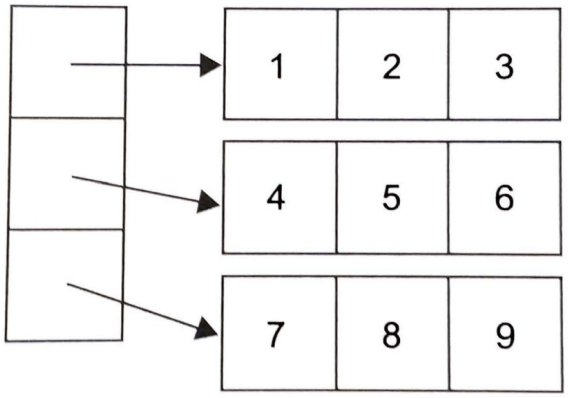


*Figura 1 – Definição de array multidimensional.Fonte: SOARES, 2014.*

## Subseção A – Array Unidimensional

O diferencial do PHP para outras linguagens de programação é que ele permite que diferentes tipos de dados sejam armazenados em um único array. No exemplo abaixo, instanciamos uma variável $arr, através da função array, que recebe por parâmetros cada um dos valores a ser armazenados no array. Neste caso, o mesmo array tem armazenado um número inteiro, uma string e um booleano (SILVA, 2014).

```
<?php
   $arr = array(5,"casa",true);
?>
```

Existem, ainda, outras maneiras de se criar esse mesmo array:

1. definindo cada chave através da função array.

```
<?php
   $arr = array(0=>5,1=>"casa",2=>true);
?>
```

2. informando os valores ou especificando os valores para cada chave.

```
<?php
   $arr[] = 5;
   $arr[] = "casa";
   $arr[] = true;
   //Ou
   $arr[0] = 5;
   $arr[1] = "casa";
   $arr[2] = true;
?>
```

3. utilizar os colchetes.

```
<?php
   $arr = [5, "casa", true];
   //Ou
   $arr = [0=>5, 1=>"casa", 2=>true];
?>
```

Caso as chaves não sejam definidas, o array incrementa automaticamente as chaves para o array a partir do 0. Por serem variáveis, o array pode ter valores adicionados e removidos durante a execução.

As chaves do array podem ser representadas por números inteiros ou strings. Podemos, por exemplo, criar um cadastro para um usuário que estará dentro de um array, em que as posições do array são os dados do usuário. Utilizamos até o momento o echo para imprimir na tela os valores de variáveis; para arrays e objetos podemos utilizar o print_r.

```
<?php
   $arr["nome"] = "Felipe Perez";
   $arr["idade"] = 30;
   $arr["cidade"] = "Dourados";
   print_r($arr);
   //resultado:
//Array ( [nome] => Felipe Perez [idade] => 30 [cidade] => Dourados )
?>
```

Dessa forma, cada posição vai ser recuperada através de sua chave, que pode ter valores variados, o que faz com que a tarefa de percorrer o array não seja tão trivial. Para isso, utilizamos a estrutura estudada na aula anterior o foreach. Com ele podemos percorrer o array, mesmo sem saber suas chaves.

Podemos criar um array com o nome de estados e definir que as chaves serão as siglas e percorrer utilizando o foreach para imprimir na tela siglas e nomes de estado. Além disso, podemos criar um outro array com as capitais que também serão recuperadas pelas chaves (siglas).

```
<?php
   $arr["MS"] = "Mato Grosso do Sul";
   $arr["SP"] = "São Paulo";
   $arr["PR"] = "Paraná";

   $capital["MS"] = "Campo Grande";
   $capital["SP"] = "São Paulo";
   $capital["PR"] = "Paraná";
   foreach($arr as $sigla => $estado) {
         echo $sigla.' - '.$estado.' - capital: '.$capital[$sigla].'<br>';
   }
   //resultado:
```

```
    //MS - Mato Grosso do Sul - capital: Campo Grande
    //SP - São Paulo - capital: São Paulo
    //PR - Paraná - capital: Paraná
    ?>
```

## Subseção B – Array Multidimensional

No exemplo anterior precisamos criar 2 arrays para armazenar o nome do estado e a capital do estado. No PHP é possível desenvolver arrays com mais de uma dimensão. Silva (2014) explica ainda que nem é preciso ter a mesma dimensão em cada uma das posições. Podemos reescrever o exemplo anterior utilizando o multidimensionamento dos arrays do PHP.

```
    <?php
      $arr["MS"]["estado"] = "Mato Grosso do Sul";
      $arr["SP"]["estado"] = "São Paulo";
      $arr["PR"]["estado"] = "Paraná";

      $arr["MS"]["capital"] = "Campo Grande";
      $arr["SP"]["capital"] = "São Paulo";
      $arr["PR"]["capital"] = "Paraná";
      foreach($arr as $sigla => $estado) {
            echo $sigla.' - '.$estado["estado"].' - capital: '.$estado["capital"].'<br>';
      }
    //resultado:
    //MS - Mato Grosso do Sul - capital: Campo Grande
    //SP - São Paulo - capital: São Paulo
    //PR - Paraná - capital: Paraná
    ?>
```

Criamos, então, apenas um array que em cada posição leva a um novo array, contendo o nome do estado e a capital. Pode-se, ainda, utilizar um foreach dentro de outro para poder percorrer os arrays multidimensionais, quando for necessário.

# 2- Arquivos

Várias funções podem ser realizadas sobre arquivos pelo PHP, seja incluído em um outro arquivo, enviado via upload, ler, escrever, renomear e mover dentro do servidor. São funcionalidades importantes que podem ser muito úteis na implementação de sistemas web (SILVA, 2014).

## Subseção A – include, include_once, require, require_once

PHP (2019) trata as funções include e require (e seus complementares include_once e require_once) também como estruturas de controle. Elas têm por função inserir códigos de arquivos externos ao script que está em execução.

A diferença entre eles se dá pelo tratamento aos erros quando os arquivos são carregados, enquanto o include gera apenas um warning o require gera um erro fatal, que encerra a execução do programa (SOARES, 2019).

Silva (2014) explica, inclusive, que o arquivo a ser incluído deve ser do tipo texto e pode conter texto puro além de códigos PHP e recursos web como HTML, CSS e JS. E a diferença entre as funções e seus complementares trata somente da verificação se o arquivo já foi incluído anteriormente na mesma execução. O include_once e o require_once, caso já tenham incluído o arquivo anteriormente, não faz nova inclusão.

Essa inclusão faz com que tudo que está dentro do arquivo seja copiado, sejam comandos, variáveis ou até saídas de tela (SILVA, 2014).

Assim como no arquivo principal, os arquivos inseridos devem seguir as regras do PHP. Tudo que não for delimitado com as tags de abertura e fechamento do PHP são considerados HTML (SOARES, 2014).

```
    <?php
      //variaveis.php
      $ano["meses"] = array("Jan", "Fev", "Mar", "Abr", "Mai", "Jun", "Jul", "Ago", "Set", "Out", "Nov", "Dez");
    $ano["dias"] = array("Seg", "Ter", "Qua", "Qui", "Sex", "Sab", "Dom");?>
    ?>

    <?php
      //index.php
      print_r($ano["dias"]);
      include("variaveis.php");
      print_r($ano["dias"]);
    print_r($ano["meses"]);
    ?>
```

O exemplo acima cria dois arquivos, o variaveis.php e o index.php. No primeiro foi criado e inicializado o array $ano, que possui os meses e dias do ano. No segundo, antes de incluir o arquivo variaveis.php, ele tenta imprimir o array, mas ainda não foram incluídas as variáveis, que serão apresentadas depois do include.

## Subseção B – Manipulação de arquivos

Assim como em outras linguagens de programação, o PHP fornece funções que possibilitam o gerenciamento de arquivos, seja para armazenar parâmetros (arquivos .ini), de configuração, arquivos-texto ou binários (SOARES, 2014).

As funções que serão apresentadas são para manipular arquivos binários, visto que essas mesmas funções podem ser utilizadas para manipular arquivos-texto, porém as funções do PHP criadas para manipular arquivos-texto não podem manipular arquivos binários.

**fopen**

Essa função é responsável por abrir o arquivo, o nome do arquivo e o modo de abertura são passados como parâmetros. (W3SCHOOLS, 2019).

Para exemplificar, utilizaremos o exemplo disponível em W3schools (2019), que cria um arquivo de texto "webdictionary.txt" com o seguinte conteúdo:

AJAX = Asynchronous JavaScript and XML

CSS = Cascading Style Sheets

HTML = Hyper Text Markup Language

PHP = PHP Hypertext Preprocessor

SQL = Structured Query Language

SVG = Scalable Vector Graphics

XML = EXtensible Markup Language

***Conteúdo arquivo webdictionary.txt. Fonte: W3schools, 2019.***

Caso a função fopen não consiga abrir o arquivo que foi passado por parâmetro, a função retornará falso. O usuário em que o servidor está rodando deve ter permissão para acessar o arquivo para poder manipulá-lo (SOARES, 2014).

Silva (2014) apresenta os tipos mais usados de abertura de arquivo texto, que devem ser passados como parâmetro, juntamente com o nome do arquivo. São eles:

| Tipo | Descrição |
|---|---|
| r | Apenas leitura, o ponteiro aponta para o início do arquivo. |
| w+ | Leitura e escrita, apaga o conteúdo do arquivo se ele existir, se não existir cria um novo, o ponteiro aponta para o início do arquivo. |
| a+ | Leitura e escrita, não apaga o conteúdo do arquivo e posiciona o cursor no final do arquivo. |

```
<?php
   $myfile = fopen("webdictionary.txt","r") or die("Incapaz de abrir o arquivo");
   ?>
```

No exemplo acima, uma variável é instanciada com o resultado da tentativa de abrir o arquivo. Caso não seja possível abri-lo, o programa é encerrado (die), a função retorna falso e a string é exibida "Incapaz de abrir o arquivo". Caso a abertura seja feita corretamente, é com essa variável que será possível fazer as manipulações posteriores com o arquivo.

**fclose**

Essa função é chamada, assim que não for mais necessária a manipulação do arquivo. É uma boa prática de programação fechar todos os arquivos, assim que tiverem sido utilizados. (W3SCHOOLS, 2019).

Para fechar o arquivo, basta passar para a função fclose, o arquivo que deve ser fechado.

```
<?php
   $myfile = fopen("webdictionary.txt","r") or die("Incapaz de abrir o arquivo");
   fclose($myfile);
   ?>
```

**fread**

A função fread faz a leitura de um arquivo que esteja aberto. Por parâmetro ela recebe a variável que foi aberta com o fopen e, em seguida, a quantidade de bytes que será lida ou até que o fim do arquivo seja encontrado.

```
<?php
   $myfile = fopen("webdictionary.txt","r") or die("Incapaz de abrir o arquivo");

fread($myfile,filesize("webdictionary.txt"));
   fclose($myfile);
   ?>
```

**fgets e feof**

As funções fgets e feof podem ser utilizadas em conjunto para fazer a leitura linha a linha do arquivo. Enquanto fgets lê apenas uma linha do arquivo (dependendo do sistema operacional o fim da linha será representado por \r ou \r\n ou \n), que pode ser armazenada em uma variável ou exibida na tela, o feof informa quando o cursor de leitura estiver no fim do arquivo.

```
<?php
   $myfile = fopen("webdictionary.txt","r") or die("Incapaz de abrir o arquivo");
   while(!feof($myfile)){
         echo fgets($myfile);
}
   fclose($myfile);
   ?>
```

**fwrite**

Essa função é utilizada para escrever no arquivo. Ela recebe por parâmetros a variável aberta desse arquivo e a string que será escrita no arquivo. Para exemplificar, abriremos um arquivo novo com o modo "w" para que, caso o arquivo não exista, ele seja criado, mas caso exista, ele exclua o conteúdo do arquivo.

O ponteiro será colocado no início do arquivo, onde armazenaremos duas strings. Por fim, o arquivo será fechado.

```
<?php
   $myfile = fopen("newfile.txt","w") or die("Incapaz de abrir o arquivo");
   $s = "Felipe Perez\n";
   fwrite($myfile,$s);
   $s = "PHP developer\n";
   fwrite($myfile,$s);
   fclose($myfile);
   ?>
```

## Retomando a aula

Chegamos ao final da segunda aula. Vamos recordar?

**1 – Arrays**

Na primeira seção foi detalhado o funcionamento dos arrays, unidimensionais e multidimensionais, desde sua inicialização à recuperação do conteúdo. Resta ressaltar que o acesso às posições pode se dar pelas chaves criadas automaticamente ou setadas pelo programador.

**2 – Arquivos**

Na segunda seção, o gerenciamento de arquivos foi exemplificado, como incluir um arquivo dentro de um script PHP, além de criar, editar, ler e fechar arquivos binários externos.

## Vale a pena

### Vale a pena **ler**

SOARES, W. *Programação Web com PHP 5*, Érica, 2014.

BEIGHLEY, L; MORRISON, M. *Use a Cabeça!* PHP & MySQL. Alta Books, 2010.

SILVA, J. M. C. *PHP na prática 200 Exercícios resolvidos*. Elsevier, 2014.

BENTO, E. J. *Desenvolvimento web com PHP e MySQL*. Editora Casa do Código, 2014.

### Vale a pena **acessar**

PHP. In: PHP Documentation, 2019. Disponível em: https://www.php.net/docs.php. Acesso em: 20 out. 2019.

W3SCHOOLS. In: PHP File Open/Read/Close, 2019. Disponível em: https://www.w3schools.com/php/php_file_open.asp. Acesso em: 21 out. 2019.

## Minhas anotações