

PHP - Introdução

Arrays

É um tipo de variável que possibilita armazenar vários valores em uma única variável.

O exemplo a seguir mostra como usar uma lista de valores em uma única variável.

Arquivo index.php

```
<?php
    $ingredientes = ['farinha', 'leite', 'açúcar', 'nata', 'fermento', 'ovos'];
    echo $ingredientes[0];

?>
```

Observem que foi declarado o array de nome ingredientes e dentro de colchetes cada valor vai separado por vírgula.

Na linha do comando echo \$ingredientes[0]; faz com que mostre na tela o valor da primeira posição no array, ou seja, farinha.

Se quiser pegar o quarto valor dentro neste array, o comando para isso seria echo \$ingredientes[3].

Para criar um array não precisa deixar o comando em uma única linha, por exemplo, pode ser feito assim:

```
$ingredientes = ['farinha',
                'leite',
                'açúcar',
                'nata',
                'fermento',
                'ovos'];
```

Vai funcionar da mesma forma, e fica melhor a visualização. Enquanto não houver o ponto e vírgula (;) o php entende que é um único comando.

Operador Array Spread(7.4)

Recurso a partir do php 7.4, portanto para funcionar precisa essa versão do php 7.4 instalado no computador pessoal.

A seguir arquivo index.php, mostra o uso desse operador spread. Por exemplo, em duas receitas de bolo (bolo1 e bolo2), não queremos repetir no bolo2 o, ou os ingredientes que já existe no array bolo1. O bolo2 é diferente em apenas um ingrediente que é o corante. Para isso na variável bolo2, colocamos o operador spread (...nomeVariavel), ou seja, três pontos seguidos do nome da variável que se quer utilizar seus valores que nesse exemplo a bolo1.

```
$bolo1 = ['farinha',  
         'leite',  
         'açucar',  
         'nata',  
         'fermento',  
         'ovos'  
        ];
```

```
$bolo2 = [  
        ...$bolo1,  
        'corante'  
        ];
```

```
echo $bolo2[3];
```

Se nessa variável \$bolo2, acrescentar mais ingredientes antes do operador spread conforme exemplo a seguir:

```
$bolo2 = [  
        'raspa de laranja',  
        'agua norma',  
        ...$bolo1,  
        'corante'
```

```
];
```

```
echo $bolo2[0];
```

Assim vai mostrar como saída o elemento na posição 0 do array \$bolo2 que é raspa de laranja.

Com o operador spread mais a variável (...\$bolo1), vai pegar, juntar, esses ingredientes, valores no array bolo2, ou seja, faz a junção dos dois arrays.

O operador spread pode ser usado diversas vezes.

Outro exemplo:

```
$lista1 = ['Sandra','Fernanda','Carolina','Thais'];
```

```
$lista2 = ['Alan','Marco','Edson','Fabiano'];
```

```
$lista3 = [...$lista1, ...$lista2];
```

```
print_r($lista3);
```

Na variável \$lista3 foi feita a junção dos dois primeiros array.

DICA: Para visualizar tudo que tem dentro de um array pode-se usar o comando print_r(). Mostra os índices e os elementos associados no array especificado.

Outro exemplo:

```
$lista1 = ['Sandra','Fernanda','Carolina','Thais'];
```

```
$lista2 = ['Alan','Marco','Edson','Fabiano'];
```

```
$lista3 = [...$lista1, 'Antônio', ...$lista2];
```

```
print_r($lista3);
```

Na variável \$lista3 foi adicionado mais um valor a esse array.

Lembrando que o operador spread é um recurso disponível a partir do PHP versão 7.4.

Estruturas de Controle

Comandos de condição

Permite executar comandos ou blocos de comandos com base em testes feitos durante a execução.

Os mais comuns: IF e SWITCH

If - else

- If: testa uma condição e executa o comando indicado se o resultado for true.

Se for necessário incluir mais de um comando no if, utilize um bloco demarcado por chaves { }.

- O else é um complemento opcional para o if. Será executado se a expressão retornar o valor false.

Também utilize { } para incluir mais do que 1 comando relacionado com o else.

If – Else - SINTAXE

If(condição) comando;	If(condição){ comando1; comando2; ... }	If(condição) comando1; else comando2;	If(condição){ comando1; ... }else{ comando3 ... }
--------------------------	---	--	---

- Você pode encadear vários ifs utilizando o comando ELSEIF:

```
if (expressão)
```

```
    comando1;
```

```
elseif (expressão2)
```

```
    comando2;
```

```
elseif(expressão3)
    comando3;
```

Switch

- É semelhante ao if, mas pode testar várias condições "exatas".

O comando break serve para evitar que as demais expressões sejam avaliadas.

```
switch (operador) {
    case valor1 : comandos; break;
    case valor2 : comandos; break;
    case valor3 : comandos; break;
    ...
    default: comandos; break;
}
```

Exemplo:

Vamos supor que estamos fazendo um feed do facebook. Para isso temos que ter algo similar à foto, vídeo e texto. Observem vários ifs no código, que dependendo do valor da variável vai ser apresentado uma determinada informação.

```
<?php
    $tipo = 'foto';
    if($tipo == 'foto'){
        echo 'Exibindo uma foto';
    }
    if($tipo == 'video'){
        echo 'Exibindo uma vídeo';
    }
    if($tipo == 'texto'){
        echo 'Exibindo uma texto';
    }
}
```

```
}  
?>
```

Quando isso ocorre é aconselhável usar o comando SWITCH. O switch é utilizado quando se tem um valor e vários resultados.

```
<?php  
    $tipo = 'texto';  
    switch($tipo){  
        case 'foto':  
            echo 'Exibindo uma foto';  
            break;  
        case 'video':  
            echo 'Exibindo uma vídeo';  
            break;  
        case 'texto':  
            echo 'Exibindo uma texto';  
            break;  
    }  
?>
```

Exemplo:

Faça um programa em PHP que, dada à idade de um aluno, verifica se o mesmo é maior de idade.

```
<?php  
    $idade = 40;  
    if($idade > 18){  
        echo "Maior de idade";  
    }else{  
        echo "Menor de idade";  
    }
```

```
}  
?>
```

Operador Ternário

Operador ternário ou operador condicional ternário, significa **if** de uma linha.

Sintaxe:

(condição) ? resultado positivo : resultado negativo;

Conforme a sintaxe dentro de parênteses, vai ter a condição e logo depois o sinal de exclamação significando que é um operador ternário seguido do resultado positivo ou negativo.

Como funciona na prática?

```
$idade = 40;
```

```
echo ($idade > 18) ? 'Maior de idade' : 'Menor de idade';
```

Também pode ser armazenado em uma variável:

```
$resultado = ($idade > 18) ? 'Maior de idade' : 'Menor de idade';
```

Isso significa que a variável resultado vai armazenar Maior de idade ou Menor de idade.

Armazenar um boolean:

```
$maiorDedade = ($idade > 18) ? true : false;
```

CondicionaL NULL CAO(7.4)

É uma simplificação da condicional ternário. Funciona somente com a versão PHP 7.4.

```
<?php
    $nome = 'Sandra';
    $nomeCompleto = $nome;
    $nomeCompleto .= $sobrenome;
    echo $nomeCompleto;
?>
```

Ao executar esse código acima vai dar um erro:

Notice: Undefined variable: sobrenome, ou seja, variável indefinida.

Como evitar que dê um erro desses?

Com o **operador ternário** tem como prevenir esse erro. Para isso faz uma condicional. Veja modificação a seguir:

```
<?php
    $nome = 'Sandra';
    $nomeCompleto = $nome;
    $nomeCompleto .= ($sobrenome) ? $sobrenome : "";
    //No lugar de aspas ( " ) pode-se usar NULL
    echo $nomeCompleto;
?>
```

Observem a condicional, se o sobrenome existir põem o sobrenome, caso contrário não põem nada. Mas mesmo assim ainda vai dar erro. Então para isso não ocorrer temos que colocar a função **isset()**, verifica se a variável é definida, se está setada, ou seja, se existe. Veja aplicação dessa função no código abaixo:

```
<?php
    $nome = 'Sandra';
    $nomeCompleto = $nome;
    $nomeCompleto .= isset($sobrenome) ? $sobrenome : "";
    echo $nomeCompleto;
?>
```


Como podem ver executando o código acima funciona normal sem erro. Mas a partir do PHP 7.4 tudo isso que foi feito anteriormente pode ser simplificado com a condicional **NULL CAO**.

Exemplo:

Essa mesma linha do código anterior com a condicional

```
$nomeCompleto .= isset($sobrenome) ? $sobrenome : '';
```

pode ser simplificado:

```
$nomeCompleto .= $sobrenome ?? '';
```

ou seja **SINTAXE:**

```
$nomeCompleto .= $sobrenome ?? resultado negativo
```

Se a variável \$sobrenome existe ela pega a própria variável, caso contrário não apresenta nada.

```
<?php
```

```
    $nome = 'Sandra';
```

```
    //$sobrenome = 'Amorim';
```

```
    $nomeCompleto = $nome;
```

```
    $nomeCompleto .= $sobrenome ?? '';
```

```
    echo $nomeCompleto;
```

```
?>
```

Como podem perceber no código anterior, sabemos que usa uma condicional de NULL CAO pelos dois operadores de exclamação (??).

Resumo:

Se quiser saber, se a variável existe com o operador ternário coloco:

```
... ? ... : ....
```

Se ela existir, uso ela mesma, caso contrário uso outra coisa.

Usando o operador ternário estamos repetindo essa variável. Para não haver essa repetição usamos o operador NULL CAO.

... ?? ...

Outro exemplo:

```
<?php
    $nomeCompleto .= $nome ?? 'Visitante';
    $nomeCompleto .= $sobrenome ?? " ";
    echo $nomeCompleto;

?>
```

Observem nesse exemplo anterior não temos nem nome e nem sobrenome e aí mostra pelo menos Visitante.

Se criarmos a \$nome = "Sandra" por exemplo mostrará Sandra no lugar de Visitante.

No site abaixo tem toda a documentação do PHP

https://www.php.net/manual/pt_BR/index.php

ou

[php.net](https://www.php.net)

Estruturas de Controle

Laços de Repetição

Se houver algo que você precisa preparar da mesma maneira muitas vezes, utilize um LOOP para repetir algumas partes do seu programa.

- WHILE
- DO WHILE
- FOR

Loop While

Testa uma condição e executa um comando, ou um bloco de comandos, até que a condição testada seja falsa.

Se no teste inicial a condição for avaliada como false, o bloco de comandos não será executado.

While(condição) comando;	while(condição){ comando1; comando2; ... }	While(): comando1; comando2; ... Endwhile
-----------------------------	--	---

Fazer um programa em PHP para imprimir de 1 a 10, usando os operadores de pós-incremento.

- Imprimir os números na forma N: 1, N:2 ... N: 10

Exemplo:

```
<?php
    $numero = 1;
    while($numero <= 10){
        echo "N: ".$numero."<br>";
        //$numero = $numero + 1;
        //$numero += 1;
        $numero++;
    }
?>
```

Do... while

O laço do...while funciona de maneira bastante semelhante ao while, com a simples diferença que a condição é testada ao final do bloco de comandos.

Isso significa que a instrução dentro do bloco é sempre executada, pelo menos uma vez.

```
do{
    comando1;
    comando2;
    ...
}
```

```
}while(condição)
```

Exemplo:

```
<?php
    $numero = 1;
    do{
        echo "N: ".$numero."</br>";
        $numero++;
    }while($numero <= 10)
?>
```

Outro exemplo:

Script em PHP utilizando laço de repetição do... while calcula e exibe todos os números pares no intervalo de 1 a n.

```
<?php
    $n=20;
    $cont=1;
    do{
        if (($cont%2)==0)
            echo $cont."</p>";
        $cont++;

    }while ($cont<=$n);
?>
```

Loop For

É o tipo de laço de repetição mais complexo.

- **É composto de:**

- **inicialização:** comando ou sequência de comandos a serem realizados antes do início do laço.

- **condição:** expressão booleana que define até quando os comandos que estão no laço serão executados.

Enquanto a expressão for verdadeira os comandos serão executados.

- **incremento:** Comando executado ao final de cada execução do laço. Pode ser um valor positivo (incremento) ou negativo (decremento).

```
for(inicialização; condição; incremento)
    comando;

ou

for(inicialização; condição; incremento){
    comando1;
    comando2;
    ...
}
```

Fazer um programa em PHP para imprimir de 1 a 10, usando os operadores de pós-incremento.

- Imprimir os números na forma N: 1, N:2 ... N: 10

Exemplo:

```
<?php
    for($numero = 1;$numero <=10; $numero++){
        echo "N: ".$numero."<br>";
    }
?>
```

Outro exemplo:

```
<?php
```

```
$n=20;

for ($cont=2; $cont<=$n; $cont+=2)

echo $cont.'</p>';

?>
```

Observação: Geralmente se usa a variável auxiliar no comando for com apenas uma letra, exemplo:

```
for($i=1;$i<=10;$i++){

    echo $i.'</br>';

}
```

Loop Foreach

Loop que se usa muito no php, feito para um único propósito que é trabalhar com arrays.

For each = para cada

Sintaxe:

```
foreach(array as variavel){

}
```

Esse comando faz uma varredura no array.

Exemplo:

?php

```
$ingredientes = [

    'farinha',

    ' essência de baunilha ',

    'leite',

    'ovos',

    'fermento'

];

echo '<h2> Ingredientes </h2>';
```

```
//para exibir os ingredientes desse array usamos o comando foreach  
foreach($ingredientes as $valor){  
    echo "Item: ".$valor."<br>";  
}  
?>
```

Observem, com o comando foreach, foi feito loops no array ingredientes e em cada loop houve armazenamento na variável \$valor.

Para pegar a **chave de identificação** de cada itens no array ingredientes pode ser feito com o comando **print_r()** ou **foreach**.

Por exemplo, se tirarmos o comando foreach do exemplo anterior e colocar print_r() mostrará a chave associada a cada valor no array:

```
print_r($ingredientes);
```

Mostra assim:

```
Array ( [0] => farinha [1] => essência de baunilha [2] => leite [3] => ovos [4] => fermento )
```

No comando foreach declaramos outra variável seguida de " => " que pega a chave associada a cada valor no array, exemplo:

```
foreach($ingredientes as $chave => $valor){  
    echo "Item ".$chave.": ".$valor."<br>";  
}
```

Após salvando e executar o código mostrará assim na tela do navegador:

Item 0: farinha

Item 1: essência de baunilha

Item 2: leite

Item 3: ovos

Item 4: fermento

Podemos ainda mudar essa listagem na numeração, o número no array começa do zero, para isso basta mudar o código dessa forma:

```
foreach($ingredientes as $chave => $valor){  
    echo "Item ".$chave + 1).": ".$valor."<br>";  
}
```

Ficando assim:

Item 1: farinha

Item 2: essência de baunilha

Item 3: leite

Item 4: ovos

Item 5: fermento

Se quisermos mostrar os ingredientes na forma de **lista não ordenada**, basta mudar para forma que está no código a seguir:

```
echo '<ul>';  
foreach($ingredientes as $valor){  
    echo '<li>'.$valor.'</li>';  
}  
echo '</ul>';
```

Resultado dessa mudança no código fica assim:

- farinha
- essência de baunilha

- leite
- ovos
- fermento

Com o foreach, possibilita tornar dinâmico, fará o número de loops que for necessário.

Exercícios práticos:

Exercício 1: Fazer em linguagem PHP com que apareça o resultado conforme imagem abaixo. São ao total 10 tracinhos e 10 linhas na tela, cada linha com 10 tracinhos(-). Mas no seu código deve ter somente um tracinho desses, ou seja, um echo ' - ' .

```
-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----  
-----
```

Exercício 2: Faça com que o código a seguir funcione, sem dar nenhum erro no executar o código.

```
<?php
```

```
    $lista = [  
        'nome' => 'Sandra',  
        'idade' => 70,  
        'atributos' => [  

```

```
        'força' => 60,  
        'agilidade' => 80,  
        'destreza' => 40  
    ],  
    vida = 1000  
];  
echo "Nome: ".$lista['nome']."<br>";  
echo "Força: ".$lista['atributos']['força']."<br>";  
echo "Vida: ".$lista['vida'];  
?>
```