4. Tipos de Dados

O PHP utiliza checagem de tipos dinâmica e fraca. Dinâmica porque uma variável pode conter valores de diferentes tipos em diferentes momentos da execução do script, e fraca porque a linguagem pode realizar conversões automaticamente entre tipos diferentes de dados. O PHP suporta 10 tipos primitivos. Desses, 4 são escalares, 4 compostos e 2 especiais.

Tipos Escalares

Uma variável **escalar** é um valor simples como um int, ponto flutuante (float), string, boolean. Esse **tipo** pode ser acessado/manipulado sem nenhuma instrução adicional.

Tipos Compostos

O PHP suporta quatro tipos compostos, que são chamados assim pois são basicamente recipientes para os outros tipos de dados. Como tipo composto o PHP tem o array, Object (a partir do PHP 7.2), callables (a partir do PHP 5.4) e iterables (a partir do PHP 7.1).

Array

Array é um tipo de variável largamente usado. Consiste basicamente num conjuntos de variáveis com um indexador e um valor.

Para **declarar um array em PHP** utilizamos o construtor de linguagem `array()` , para o qual podemos passar por parâmetro os valores que desejamos armazenar, separados por vírgula, como mostra o quadro 9.

Quadro 9 – Exemplo de declaração de array

```
$array = array(1, 2, 3);  ou  $array = array[1, 2, 3];
```

O PHP permite ainda a declaração de arrays associativos. Para esse fim, o construtor array() pode receber quais serão as chaves às quais os valores estão associados como parâmetro. Um exemplo dessa sintaxe pode ser visto no quadro 10.

Quadro 10 – Exemplo de declaração de array associativo

```
$array = array(
   “chave1” => 1,
   “chave2” => “Pedro II”,
   “chave3” => false
 );
```

Para acessar um array basta digitar o nome da variável array e o seu índice com ilustrado no quadro 11.

Quadro 11 – Exemplo de como acessar um array

```
echo $array[0];
echo $array[1];
echo $array[2];
```

Também é possível acessar um valor diretamente através da chave a qual ele está relacionado. O trecho de código no quadro 12 imprime o valor Pedro II, contido na segunda posição do array para o qual a chave é chave2.

Quadro 12 – Código que imprime o valor contido na chave2

```
echo $array[“chave2”];
```

O exemplo de código no quadro 13 imprime todas as chaves do array $array, bem como o valor associado a cada uma delas:

Quadro 13 – Exemplo de código que ler todo o array

```
foreach ($array as $chave => $valor){
   echo “{$chave}: {$valor}\n“;
}
```

Object

Objeto é um tipo de variável exclusivo de programação orientada a objeto. É algo parecido com um array. Só que um objeto é composto de métodos e propriedades. Esses métodos e propriedades são determinados em classes. Um objeto é a instância de uma classe, ou seja, é a classe pronta para ser usada. O quadro 14 mostra um exemplo de uma classe chamada Fruit e de dois objetos um chamado $apple e o outro $banana.

Quadro 14 – Exemplo de classes e objetos

```
<?php
class Fruit {
  // Properties
  public $name;
  public $color;
  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}
//Objetos
$apple = new Fruit();
$banana = new Fruit();
$apple->set_name('Apple');
$banana->set_name('Banana');

echo $apple->get_name();
echo "<br>";
echo $banana->get_name();
?>
```

Fonte w3schools

Callables

Introduzida no PHP a partir da versão 5.4.0, a palavra callable é utilizada para verificar se o parâmetro é invocável. Na prática ele é usado para verificar se podemos executar o parâmetro que recebemos como uma função. O quadro 15 ilustra alguns exemplo.

Quadro 15 – Exemplos de Callable

```
//Função que recebe um invocável
function executar(callable $executavel) {
   echo $executavel();
}

$funcao = function() {
   echo "executou a função<br>";
};


//passa a função que está dentro da variável $funcao
executar($funcao);

//passa uma função anônima
executar(function() {
   echo "executou a função<br>";
});
function funcao() {
   echo "executou a função funcao<br>";
};

//passa o nome da função
executar("funcao");

class ClasseExecutavel {
   public function __invoke() {
      return "executou a função dentro da classe<br>";
   }
}
//passa a instancia de uma classe executável
executar(new ClasseExecutavel());
```

Iterable

O desenvolvedor PHP, está acostumado a usar arrays para trabalhar e iterar sobre listas. Mas também é possível encapsular esse tipo de comportamento em um objeto. No contexto prático do PHP, *Iterator* é um mecanismo que permite que um objeto seja iterado e ele próprio fica no controle granular dessa iteração. Mas o seu uso não limita a essa "característica".

Então, de forma prática, tudo o que você pode iterar num `foreach` é considerável iterável. O quadro 16 mostra um exemplo.

Quadro 16 – Exemplo de código com foreach

```
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
    echo "$value <br>";
}
?>
```

Fonte w3scholl

Propriedades tipadas

O PHP 7.4 lançado no final de 2019 implementou uma grande novidade para a linguagem, o suporte a propriedades tipadas. Esses recursos em conjunto com todos os outros que já vistos permite um controle ainda maior de tipos. Mas antes de ir aos exemplos, será preciso entender um outro conceito, o `strict_types`.

Por padrão todos os arquivos PHP estão em modo *"fraco"* de checagem de tipo, para entender melhor veja o exemplo do quadro 17.

Quadro 17 – Exemplo de código de tipagem fraca

```
<?php
function sum(int $num1) : int
{
   return $num1 + 1;
}
var_dump(sum('1')); // int(2)
```

Perceba que a função foi assinada para receber um parâmetro $num1 do tipo int, e retornar também um int, sendo a soma do parâmetro passado mais 1, porém a chamada da função é feita passando a string '1' e mesmo assim o PHP conseguiu interpretar, converter (string -> int) e retornar um int, que neste caso foi o número 2. E isso só foi possível devido o código estar com o modo fraco de tipo de checagem.
Agora com o modo strict_types ativado e com o mesmo exemplo do quadro anterior.

Quadro 18 - Exemplo de código de tipagem fraca com strict_types

```
<?php
  declare(strict_types=1);

  function sum(int $num1) : int
  {
     return $num1 + 1;
  }
  var_dump(sum('1'));
```

A única mudança feita foi a inclusão do comando declare(strict_types=1), ou seja, a partir deste momento, a checagem de tipo será feita, e é como se estivesse dizendo para o PHP para utilizar o modo "forte" de checagem de tipo. O exemplo do quadro 18 irá produzir um fatal error, como abaixo:

> *PHP Fatal error: Uncaught TypeError: Argument 1 passed to sum() must be of the type integer, string given*

Agora que já sabemos como utilizar o modo "forte" de checagem de tipo do PHP, vamos ver quais são os 4 tipos de dados adicionados e que agora são permitidos como declaração de tipo de parâmetro de funções ou métodos. O quadro 19 mostra um exemplo dessas declarações.

Quadro 19 – Declaração de variáveis tipadas

```
<?php
   string $nome,
   int $quantidade,
   float $valor,
   bool $entregar
?>
```

Veja o exemplo do quadro 20.

Quadro 20 – Outro exemplo de código com uso do strict_type

```
<?php

declare(strict_types=1);

class Seller
{
   private $salary = 2500;
   private $name = João Silva';

   public function getNameWithPrefix(string $sufixForName) : string
   {
         return $sufixForName . ' ' . $this->name;
   }

   public function getSalaryWithCommission(float $v) : float
   {
      return $this->salary * $v;
   }
}

$actor = new Vendedor();

var_dump($actor->getSalaryWithCommission(1.3)); // float(3250)
var_dump($actor->getNameWithPrefix('Mr.')); // string(16) "Mr. João Silva"
```

Vejam que acima, temos uma classe Seller e dois métodos getNameWithPrefix (string $sufixForName) : string e getSalaryWithCommission(float $v) : float, perceba que cada uma delas declara um parâmetro string e float respectivamente, e logo após instanciar a classe, foi feita as chamadas a estes métodos passando os parâmetros corretamente. Porém, vamos fazer uma chamada ao método getSalaryWithCommission (float $v) : float passando como parâmetro um int conforme mostrado no quadro 21.

Quadro 21 – Chamada ao método getSalaryWithCommission

```
var_dump($actor->getSalaryWithCommission(4)); // float(10000)
```

Isso deveria ter dado erro já que a função deveria ter recebido um float e o foi passado um int. Bem neste caso não, já que int são aceitos para argumentos float, porém o contrário não será aceito.

Perceba que este recurso será muito útil principalmente para as documentações de códigos, já que agora existe uma forma de **obrigar** o tipo de dado que é passado para uma função ou método.

Transformação de Tipos por coerção

Coerção é a conversão de um tipo em outro tipo diferente mediante operação realizada com tipos diferentes. Por exemplo:

```
$a=1;
$a = $a + "5";
```

Nesse exemplo a variável $a é numérica (integer) na primeira atribuição e string na segunda atribuição.

Transformação explicita de tipos

Feita via *typecast* (como em linguagem C)

Exemplo:

```
$a = 6;              // a é um integer(6)
$a = (float) 6;      // é um float(6.0)
```

Constantes

Uma constante é um identificador (nome) para um único valor que não se altera durante a execução de um script)

- Pré-definidas
- Definidas pelo usuário

Exemplo:

```
define("MOL", 6.14e22);
define("PI", 3.1415);
echo MOL;
echo PI;
```

O PHP tem algumas constantes muito uteis para depuração de código, como __FILE__ que contém o nome do arquivo, __LINE__ que contém o número da linha, __FUNCTION__ para o nome da função entre outras. Para mais detalhes consulte o manual do PHP no site php.net.

5. Operadores

Operadores de atribuição

Operador	Exemplo	Equivalente
=	($a=5);	
+=	$a+=5;	$a = $a + 5
-=	$a-=5;	$a = $a - 5
*=	$a*=5;	$a = $a * 5
/=	$a/=5;	$a = $a / 5
%=	$a%=5;	$a = $a % 5

Operadores de comparação

Operador	Nome	Exemplo	Resultado
==	Igual	$x==$y	Retorna verdade se $x for igual a $y
===	Idêntico	$x===$y	Retorna verdade se $x for igual a $y e se são do mesmo tipo
!=	Diferente	$x!=$y	Retorna verdade se $x for diferente de $y
<>	Diferente	$x!=$y	Retorna verdade se $x for diferente de $y
!==	Não Idêntico	$x!==$y	Retorna verdade se $x for diferente a $y e se são do mesmo tipo
>	Maior que	$x>$y	Retorna verdade se $x for maior que $y
<	Menor que	$x<$y	Retorna verdade se $x for menor que $y
>=	Maior ou igual que	$x>=$y	Retorna verdade se $x for maior ou igual que $y
<=	Menor ou igual que	$x<=$y	Retorna verdade se $x for menor ou igual que $y
< = >		$x <=> $y	Retorna um número inteiro menor, igual ou maior que zero, dependendo se $ x for menor, igual ou maior que $ y. Introduzido no PHP 7.

Operadores de incremento / decremento

Operador	Nome	Descrição
++$x	Pré-incremento	Incrementa $x em 1, então retorna $x
$x++	Pós-incremento	Retorna $x, então incrementa $x em 1
--$x	Pré-decremento	Decrementa $x em 1, então retorna $x
$x--	Pós-decremento	Retorna $x, então decrementa $x em 1

Operadores lógicos

Operador	Nome	Exemplo	Resultado
and	e	$x and $y	Verdade se $x e $y são verdadeiros
or	ou	$x or $y	Verdade se $x ou $y é verdadeiro
xor	Ou exclusivo	$x xor $y	Verdade se $x e $y são verdadeiros, mas não ambos
&&	e	$x && $y	Verdade se $x e $y são verdadeiros
||	ou	$x || $y	Verdade se $x ou $y é verdadeiro
!	não	!$x	Verdade se $x não for verdade

Operadores de string

Operador	Nome	Exemplo	Resultado
.	Concatenação	$txt1.$txt2	Concatena $txt1 e $txt2
.=	Atribuição de concatenação	$txt1 .= $txt2	Adiciona $txt2 em $txt1

Operadores de array

Operador	Nome	Exemplo	Resultado
+	União	$x + $y	União de $x e $y
==	Igualdade	$x==$y	Retorna verdade se $x e $y tem o mesmo pares chave/valor
===	Idêntico	$x===$y	Retorna verdade se $x e $y tem o mesmo pares chave/valor e a mesma ordem e o mesmo tipo
!=	Diferente	$x!=$y	Retorna verdade se $x for diferente de $y
<>	Diferente	$x!=$y	Retorna verdade se $x for diferente de $y
!==	Não Idêntico	$x!==$y	Retorna verdade se $x não for idêntico a $y

Operadores de atribuição condicional

Operador	Nome	Exemplo	Resultado
?:	Ternário	$x = expr1 ? expr2 : expr3	Retorna o valor de $x O valor de $ x é expr2 se expr1 = TRUE. O valor de $ x é expr3 se expr1 = FALSE
??	Coalescência nula	$x = *expr1* ?? *expr2*	Retorna o valor de $ x. O valor de $ x é expr1 se expr1 existe e não é NULL. Se expr1 não existir ou for NULL, o valor de $ x será expr2. Introduzido em PHP 7