

# **Curso PHP completo 85bits (Prof. Celso Araujo Fontes)**

<https://www.youtube.com/watch?v=Z1Kdt2dV3iM&list=PLhUp81I0jET71qMTWy50cpAYHrWnVnYIA>

Resumo do curso feito por Roberto Pinheiro

## **Aula 03 - Funções e Clousures**

### **Função básica**

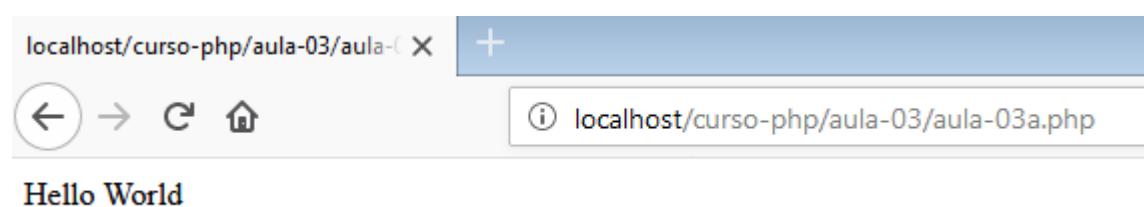
Uma função é basicamente um bloco de instruções que podem ser utilizados repetidamente num programa.

Diferente das outras instruções básicas, uma função nunca será executada imediatamente quando uma página ou script é carregado.

Uma função só será executada apenas quando é feito uma chamada pela mesma.

### **aula-03/aula-03a.php**

```
<?php  
  
function escreverMensagem(){  
    echo "Hello World";  
}  
  
escreverMensagem();  
  
?>
```



# Editor PHP online

<https://3v4l.org/>

The screenshot shows the 3v4l.org website. At the top, there is a header with the site's logo and navigation links for sponsor, bughunt, search, recent, and about. Below the header is a code editor window titled "Untitled". The code in the editor is:

```
1 <?php
2
3 |
eval();
```

There is a checkbox labeled "eol versions" to the right of the editor. Below the editor, the page displays a "Welcome to the best online PHP shell" message. It explains that 3v4l.org is an online shell for running PHP code across 300+ versions. It also lists some PHP 7.3 gotchas found through the bughunt:

- Traversables can no longer yield non-integer keys
- Instanceof now accepts non-object instances
- Pcre no longer accepts unescaped dashes in character classes
- Various errors were converted to exceptions
- Compact() now throws notices for undefined variables

On the left side of the main content area, there are two checkboxes: "Enable dark-mode" (checked) and "Enable live-preview". On the right side, there is a section titled "Popular today" which lists several RFCs.

The screenshot shows the 3v4l.org website again. This time, the code editor has a title "Função básica". The code in the editor is:

```
1 <?php
2
3 function escreverMensagem(){
4     echo "Hello World";
5 }
6
7 escreverMensagem();
```

Below the editor is a button labeled "eval();". The page then transitions to show the output for different PHP versions. The tabs at the top of this section are Output, Performance, VLD opcodes, References, and Branches. The "Output" tab is selected. The output text area contains the text "Hello World".

## Função com passagem de argumentos/parâmetros

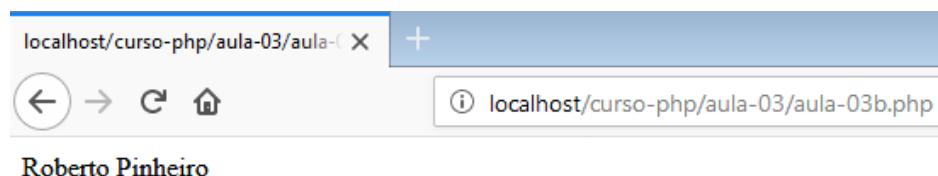
Uma informação pode ser passada para as funções por meio de argumentos. Um argumento é como uma variável (inclusive utilizamos o \$ antes do nome).

Argumentos são especificados após o nome da função, dentro dos parênteses.

Podemos adicionar quantos argumentos quisermos, basta separá-los com uma vírgula.

### **aula-03/aula-03a.php**

```
<?php  
  
function imprimeNome($nome, $sobrenome){  
    echo $nome . " " . $sobrenome;  
}  
  
imprimeNome('Roberto', 'Pinheiro');  
  
?>
```



## Função com passagem de argumentos/parâmetros

```
1 <?php
2
3 v function imprimeNome($nome, $sobrenome){
4     echo $nome . " " . $sobrenome;
5 }
6
7 imprimeNome('Roberto', 'Pinheiro');
```

eval();

Output

Performance

VLD opcodes

References

Branches

Output for 7.3.0 - 7.3.27, 7.4.0 - 7.4.16, 8.0.0 - 8.0.3

Roberto Pinheiro

## Função com um parâmetro opcional

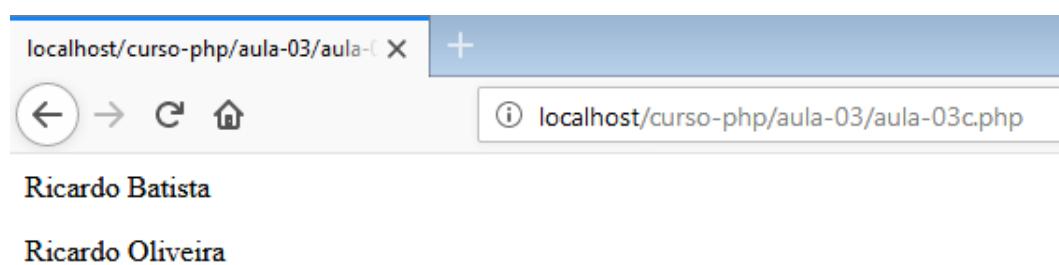
Muitas vezes é necessário que alguns parâmetros de nossa função sejam opcionais. Para isso, podemos definir valores padrões para os parâmetros que desejamos que se tornem opcionais. Esta atribuição é feita através de forma muito similar a atribuição de um valor para uma variável usando o sinal de igual seguido do valor desejado.

### aula-03/aula-03c.php

```
<?php

function imprimeNome($nome, $sobrenome = "Batista"){
    echo "<p>" . $nome . " " . $sobrenome . "</p>";
}

imprimeNome('Ricardo');
imprimeNome('Ricardo', 'Oliveira');
```



```
<?php

function imprimeNome($nome, $sobrenome = "Batista"){
    echo "$nome $sobrenome" . PHP_EOL;
}

imprimeNome('Ricardo');
imprimeNome('Ricardo', 'Oliveira');
```

## Função com um parâmetro opcional

```
1 <?php
2
3 function imprimeNome($nome, $sobrenome = "Batista"){
4     echo "$nome $sobrenome" . PHP_EOL;
5 }
6
7 imprimeNome('Ricardo');
8 imprimeNome('Ricardo', 'Oliveira');
```

eval();

Output

Performance

VLD opcodes

References

Branches

Output for 7.3.0 - 7.3.27, 7.4.0 - 7.4.16, 8.0.0 - 8.0.3

```
Ricardo Batista
Ricardo Oliveira
```

## **Lista de argumentos variável (splash operator)**

A partir do PHP 5.6 tornou-se possível incluir o token reticências para indicar uma lista de argumentos e assim informar que a função aceita um número variável de argumentos.

Os argumentos serão passados na forma de um array e por isso, neste exemplo, utilizamos uma função do php chamada array\_sum para somar os valores dentro deste array.

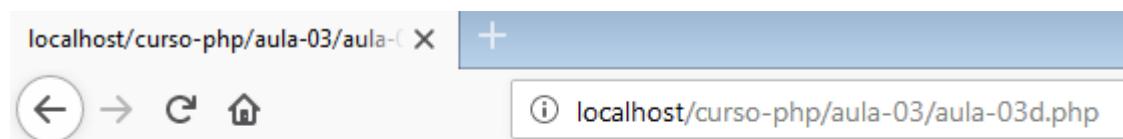
### **aula-03/aula-03d.php**

```
<?php

function valores(...$x){
    echo "<p>" . array_sum($x) . "</p>";
}

valores(143,542);
valores(1,2,3,4);

?>
```



```
<?php

function valores(...$x){
    echo array_sum($x) . PHP_EOL;
}

valores(143,542);
valores(1,2,3,4);
```

## Lista de argumentos variável (splash operator)

```
1 <?php
2
3 v function valores(...$x){
4   echo array_sum($x) . PHP_EOL;
5 }
6
7 valores(143,542);
8 valores(1,2,3,4);
```

eval();

Output

Performance

VLD opcodes

References

Branches

Output for 7.3.0 - 7.3.27, 7.4.0 - 7.4.16, 8.0.0 - 8.0.3

685  
10

## Argument unpack (splash operator)

O splat operator também pode ser utilizado para "desempacotar" uma coleção em argumentos separados.

### aula-03/aula-03e.php

```
<?php

function valores($a, $b, $c){
    echo $a + $b + $c;
}

$args = array(43, 97);
valores(10, ...$args);

?>
```



150

The screenshot shows the PHP 8.0.0 IDE interface. At the top, there's a title bar with the text "Argument unpack (splash operator)". Below it is a code editor window containing the provided PHP code. A large button labeled "eval()" is centered below the code. At the bottom of the code editor, there are tabs for "Output", "Performance", "VLD opcodes", "References", and "Branches", with "Output" being the active tab. The output panel below displays the result "150".

```
1 <?php
2
3 function valores($a, $b, $c){
4     echo $a + $b + $c;
5 }
6
7 $args = array(43, 97);
8 valores(10, ...$args);
```

eval();

Output   Performance   VLD opcodes   References   Branches

Output for 7.3.0 - 7.3.27, 7.4.0 - 7.4.16, 8.0.0 - 8.0.3

150

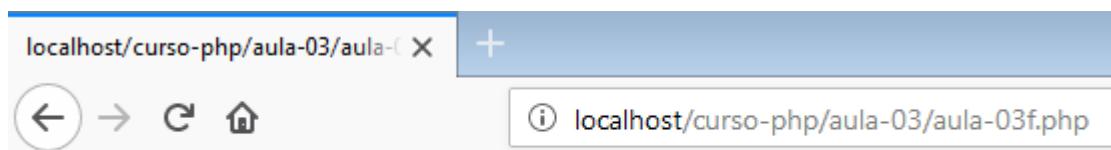
## Função com return

O return permite que uma função possa retornar algum valor.

O return retorna o controle do programa para o trecho que executou a chamada.

### aula-03/aula-03f.php

```
<?php  
  
function soma($a, $b){  
    return $a + $b;  
}  
  
$soma = soma(16, 38);  
print $soma;  
  
?>
```



```
<?php

function nomeCompleto($nome, $sobrenome = "Silva"){
    return "$nome $sobrenome";
}

$usuario = nomeCompleto('Thiago');
print $usuario;
```

Função com return

```
1 <?php
2
3 v function nomeCompleto($nome, $sobrenome = "Silva"){
4     return "$nome $sobrenome";
5 }
6
7 $usuario = nomeCompleto('Thiago');
8 print $usuario;
9
```

eval();

Output    Performance    VLD opcodes    References    Branches

Output for 7.3.0 - 7.3.27, 7.4.0 - 7.4.16, 8.0.0 - 8.0.3

Thiago Silva

## Função recursiva

### Função Recursiva

- É basicamente uma função que chama a si mesma.
- Neste exemplo utilizaremos o resultado de cada iteração para multiplicar este resultado vezes ele mesmo menos 1 em uma cadeia de recursividade que será controlada por um “if” que irá garantir que a recursividade não seja eterna.
- **Atenção:** Chamadas de função / método recursivas com mais de 100 a 200 níveis de recursão podem estourar a pilha e causar o encerramento do script atual.
- A recursão infinita é considerada um erro de programação.

### aula-03/aula-03g.php

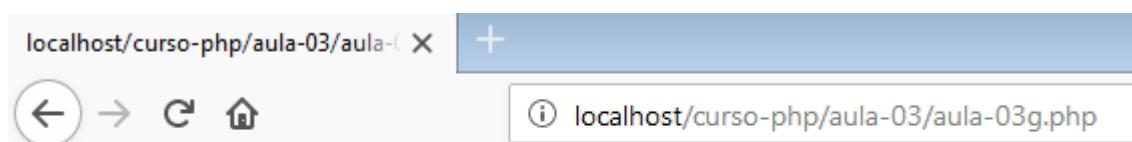
```
<?php

function fatorial($num){
    if($num == 1) {
        return 1;
    } else {
        return $num * fatorial($num - 1);
    }
}

$resultado = fatorial(4);

print $resultado;

?>
```



```
= 4 * 3  
= 4 * (3 * 2)  
= 4 * (3 * (2 * 1))  
= 4 * (3 * 2)  
= 4 * 6  
= 24
```

## Função recursiva

```
1 <?php  
2  
3 v function factorial($num){  
4 v   if($num == 1) {  
5 |     return 1;  
6 v   } else {  
7 |     return $num * factorial($num - 1);  
8 |   }  
9  
10 }  
11  
12 $resultado = factorial(4);  
13  
14 print $resultado;  
15
```

eval();

Output

Performance

VLD opcodes

References

Branches

Output for 7.3.0 - 7.3.27, 7.4.0 - 7.4.16, 8.0.0 - 8.0.3

24

## Type Hint

Tipos usados no TypeHint:

- bool
- float (php >=7)
- int (php >=7)
- string (php >=7)
- Classe/Interface
- callable
- self
- array

```
<?php
```

```
function soma(int $a, int $b) {  
    return $a + $b;  
}
```

```
$soma = soma(6.2, 8.6);  
print $soma;
```

Type Hint

```
1 <?php  
2  
3 v function soma(int $a, int $b) {  
4     return $a + $b;  
5 }  
6  
7 $soma = soma(6.2, 8.6);  
8 print $soma;  
9
```

eval();

Output    Performance    VLD opcodes    References    Branches

Output for 7.3.0 - 7.3.27, 7.4.0 - 7.4.16, 8.0.0 - 8.0.3

14

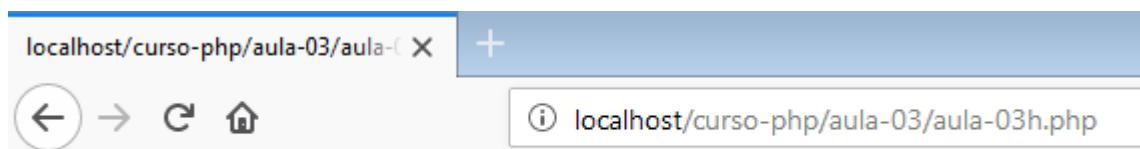
## Função (declaração de tipo de retorno: ":" )

A partir do PHP 7 também se tornou possível definir o tipo do valor do retorno de uma função.

Para utilizar este recurso, basta definir tipo da saída (retorno) antecedido pelo símbolo de dois pontos.

### **aula-03/aula-03h.php**

```
<?php  
  
function soma(float $a, float $b) : int{  
    return $a + $b;  
}  
  
$soma = soma(6.2, 8.6);  
print $soma;  
  
?>
```



14

O Type Hint ainda pode ser usado em conjunto com o Splat Operator para especificar qual o tipo dos valores de um determinado conjunto.

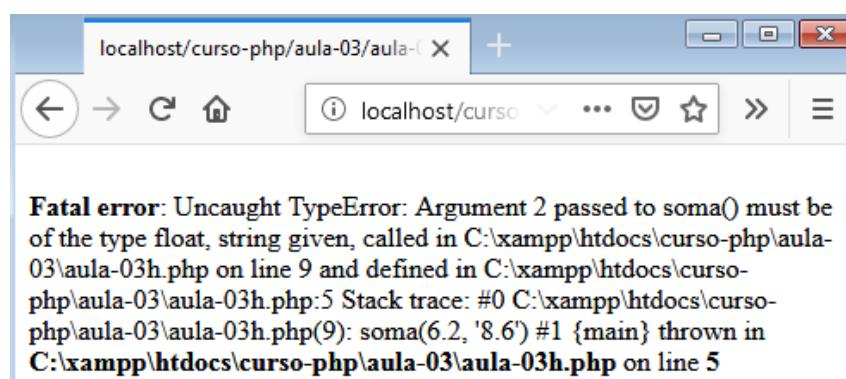
## Restrição de tipos ligada

No PHP também é possível controlar o nível de restrição do Type Hint, isto é, definir se a restrição estará ligada "1" ou desligada "0".

Vale a pena salientar que o strict\_types precisa ser a primeira instrução no script.

### aula-03/aula-03i.php

```
<?php  
  
declare(strict_types=1);  
  
function soma(float $a, float $b) : int{  
    return $a + $b;  
}  
  
$soma = soma(6.2, '8.6');  
print $soma;  
  
?>
```



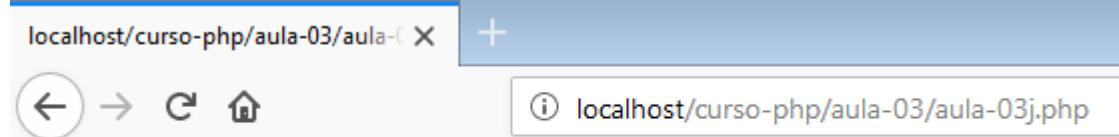
## Nullable types (php 7.1 ou superior)

Com o advento do PHP 7, foi possível determinar uma nova condição de valor para o tipo de parâmetro (ou retorno de função), usando o "nullable type".

O Nullable Type permite que além de um valor com o tipo solicitado, o parâmetro também seja capaz de receber null (ou um retorno null).

### aula-03/aula-03j.php

```
<?php  
  
declare(strict_types=1);  
  
function soma(float $a, float $b, ?float $c) : float{  
    return $a + $b;  
}  
  
$soma = soma(6.2, 8.6, null);  
print $soma;  
  
?>
```



14.8

## Passagem de parâmetro por valor

### aula-03/aula-03k.php

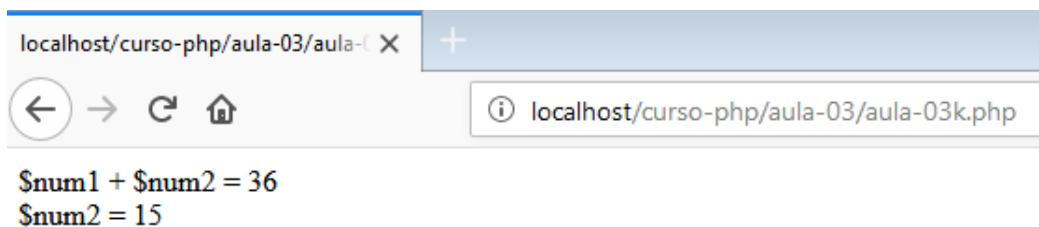
```
<?php

function soma($num1, $num2){
    $num2 = 2 * $num2;
    return $num1 + $num2;
}

$num2 = 15;

$soma = soma(6, $num2);
print '$num1 + $num2 = ' . $soma . "<br>";
print '$num2 = ' . $num2;

?>
```



## Passagem de parâmetro por referência "&"

Com o uso do e comercial é possível determinar que um parâmetro possa modificar uma variável "mantendo a referência" para a mesma dentro do escopo da variável.

### aula-03/aula-03I.php

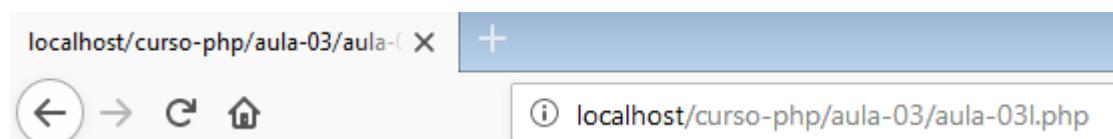
```
<?php

function soma($num1, &$num2){
    $num2 = 2 * $num2;
    return $num1 + $num2;
}

$num2 = 15;

$soma = soma(6, $num2);
print '$num1 + $num2 = ' . $soma . "<br>";
print '$num2 = ' . $num2;

?>
```



\$num1 + \$num2 = 36

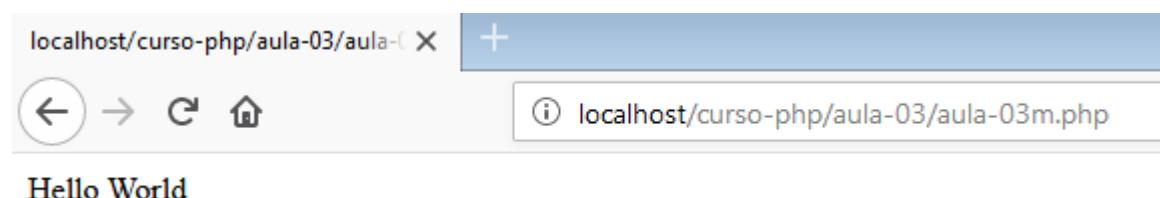
\$num2 = 30

## Função anônima ou closure

Funções anônimas, também conhecidas como closures, permitem a criação de funções que não tem o nome especificado. Elas são mais úteis como o valor de parâmetros callback, mas podem ter vários outros usos. Vale apenas salientar que **toda função anônima atribuída a uma variável termina com ponto e vírgula.**

### aula-03/aula-03m.php

```
<?php  
  
$x = function($txt){  
    echo "Hello $txt";  
};  
  
$x("World");  
  
?>
```

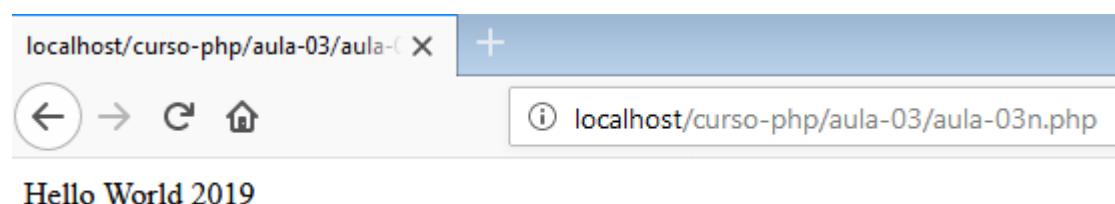


## Função anônima ("use")

O termo "use" permite a utilização de variáveis externas dentro do escopo de uma closure.

### aula-03/aula-03n.php

```
<?php  
$ano = "2019";  
  
$x = function($txt) use($ano){  
    echo "Hello $txt $ano";  
};  
  
$x("World");  
  
?>
```



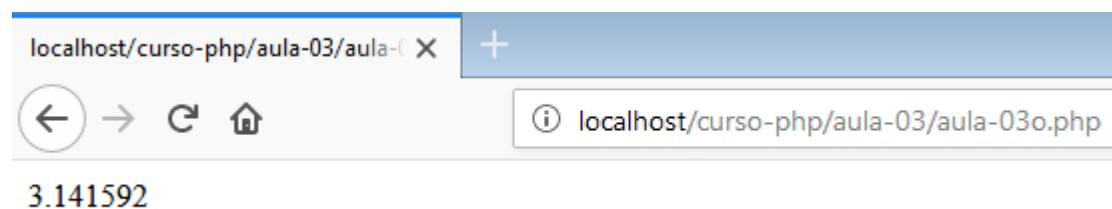
## Constantes

Diferente das variáveis, o valor de uma constante nunca muda e acessamos seu valor sem o uso do cifrão.

Para definir uma constante utilizamos a função `define`, onde o primeiro parâmetro será uma string com o nome da constante, o segundo parâmetro será o valor da constante e por último podemos definir se a constante será insensível ao case, isto é, se a constante pode ser chamada através de letras maiúsculas ou minúsculas independente de como foi definido.

### **aula-03/aula-03o.php**

```
<?php  
  
define("PI", 3.141592);  
  
echo PI;  
  
?>
```



## Função Date()

```
<?php  
date_default_timezone_set('America/Sao_Paulo');  
  
$data = date('d/m/Y H:i:s');  
  
echo $data;
```

Função date()

```
1 <?php  
2  
3 date_default_timezone_set('America/Sao_Paulo');  
4  
5 $data = date('d/m/Y H:i:s');  
6  
7 echo $data;  
8
```

eval();

Output    Performance    VLD opcodes    References    Branches

Output for 7.3.0 - 7.3.27, 7.4.0 - 7.4.16, 8.0.0 - 8.0.3

```
22/04/2021 05:41:11
```

# Aula 04 - Arrays e Funções de Array

## PHP Array

- Um array em PHP é na verdade um mapa ordenado. Um mapa é um tipo que associa **valores** a **chaves**.
- Esse tipo é otimizado para vários usos diferentes; ele pode ser tratado como uma **matriz**, **lista** (vetor), **tabela de hash** (uma implementação de um mapa), **dicionário**, **coleção**, **pilha**, **fila** e provavelmente mais.
- Como os valores de array podem ser outros arrays, árvores e arrays **multidimensionais** também são possíveis.
- Pode crescer dinamicamente e ser **iterado** em diferentes direções.

## Array - Sintaxe básica

### Varrendo um array

#### aula-04/aula-04a.php

```
<?php

$frutas = array("laranja", "banana", "maçã", "abacaxi", "uva", "melancia");

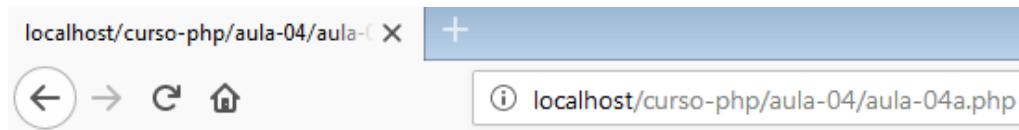
echo "<ul>";
foreach($frutas as $fruta){
    echo "<li>$fruta</li>";
}
echo "</ul>";

echo "<hr>";

$animais = ["cachorro", "gato", "cavalo", "macaco", "canguru", "elefante"];

echo "<ol>";
foreach($animais as $animal){
    echo "<li>$animal</li>";
}
echo "</ol>

?>
```



<?php

```
$frutas = array("laranja", "banana", "maçã", "abacaxi", "uva", "melancia");

foreach($frutas as $fruta){
    echo "$fruta" . PHP_EOL;
}
```

### Array - Sintaxe básica

```
1 <?php
2
3 $frutas = array("laranja", "banana", "maçã", "abacaxi", "uva", "melancia");
4
5 foreach($frutas as $fruta){
6     echo "$fruta" . PHP_EOL;
7 }
```

eval();

Output    Performance    VLD opcodes    References    Branches

Output for 7.3.0 - 7.3.27, 7.4.0 - 7.4.16, 8.0.0 - 8.0.3

```
laranja
banana
maçã
abacaxi
uva
melancia
```

```
$animais = ["cachorro", "gato", "cavalo", "macaco", "canguru", "elefante"];  
  
foreach($animais as $animal){  
    echo $animal . PHP_EOL;  
}
```

## Array - Sintaxe básica

```
1 <?php  
2  
3 $animais = ["cachorro", "gato", "cavalo", "macaco", "canguru", "elefante"];  
4  
5 foreach($animais as $animal){  
6     echo $animal . PHP_EOL;  
7 }
```

eval();

Output    Performance    VLD opcodes    References    Branches

Output for 7.3.0 - 7.3.27, 7.4.0 - 7.4.16, 8.0.0 - 8.0.3

```
cachorro  
gato  
cavalo  
macaco  
canguru  
elefante
```

## Array multidimensional

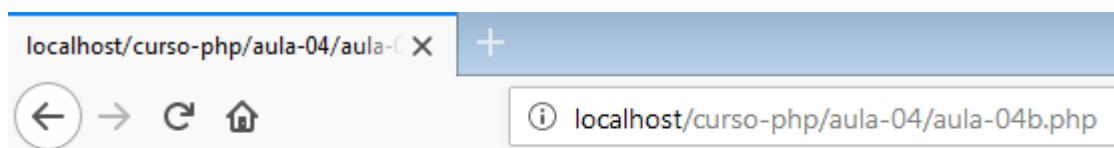
### aula-04/aula-04b.php

```
<?php

$shop = [
    ["camisa", 85.75, 12],
    ["calça", 120.15, 8],
    ["meia", 8.25, 25]
];

print $shop[0][0] . " custa R$ " . $shop[0][1] . " e possui " . $shop[0][2] . "
unidades <br>";
print $shop[1][0] . " custa R$ " . $shop[1][1] . " e possui " . $shop[1][2] . "
unidades <br>";
print $shop[2][0] . " custa R$ " . $shop[2][1] . " e possui " . $shop[2][2] . "
unidades <br>";

?>
```



## Iteração de array (foreach chave-valor)

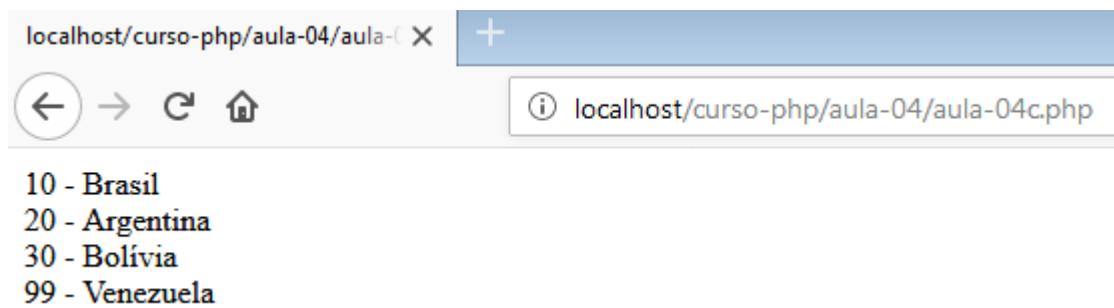
### aula-04/aula-04c.php

```
<?php

$array = [10 => "Brasil", 20 => "Argentina", 30 => "Bolívia", 99 =>
"Venezuela"];

foreach($array as $key => $pais){
    print "$key - $pais <br>";
}

?>
```



localhost/curso-php/aula-04/aula-04c.php

10 - Brasil  
20 - Argentina  
30 - Bolívia  
99 - Venezuela

## Iteração de arrays multidimensionais

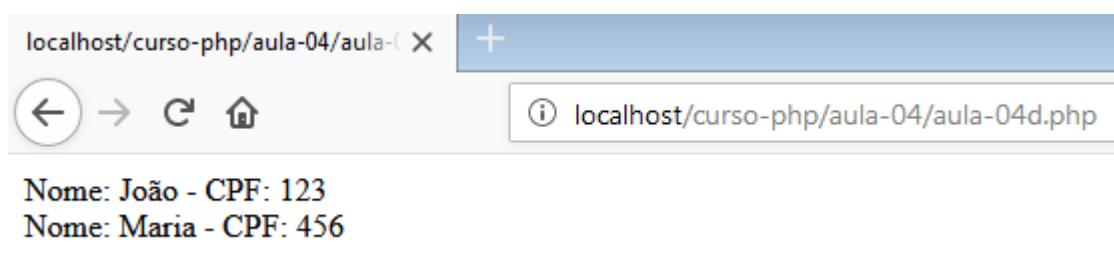
### aula-04/aula-04d.php

```
<?php

$pessoas = [
    ["nome" => "João", "cpf" => "123"],
    ["nome" => "Maria", "cpf" => "456"]
];

foreach($pessoas as $pessoa){
    print "Nome: " . $pessoa['nome'] . " - CPF: " . $pessoa['cpf'] . "<br>";
}

?>
```



localhost/curso-php/aula-04/aula-04d.php

Nome: João - CPF: 123  
Nome: Maria - CPF: 456

## Debug (print\_r, var\_dump, var\_export)

### print\_r, var\_dump, var\_export

print\_r()

```
Array
(
    [0] =>
    [1] =>
    [2] => 42
    [3] => Array
        (
            [0] => 42
        )
)
```

var\_dump()

```
array(4) {
    [0]=>
        string(0) ""
    [1]=>
        bool(false)
    [2]=>
        int(42)
    [3]=>
        array(1) {
            [0]=>
                string(2) "42"
        }
}
```

var\_export()

```
array (
    0 => '',
    1 => false,
    2 => 42,
    3 =>
        array (
            0 => '42',
        ),
)
```

### aula-04/aula-04e.php

```
<?php

$var1 = ["", "texto", 1.45, true, [1,2,3]];

var_dump($var1);

echo "<hr>";

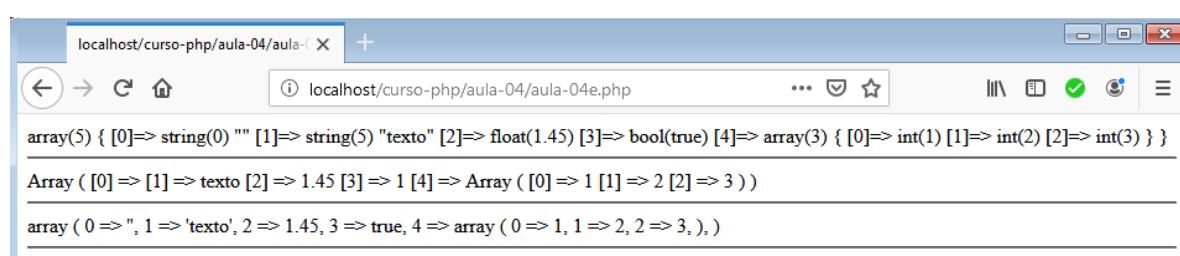
print_r($var1);

echo "<hr>";

var_export($var1);

echo "<hr>;

?>
```



## Ordenação de arrays

### aula-04/aula-04f.php

```
<?php

$frutas = ["laranja", "caqui", "goiaba", "limão", "pera", "abacate", "maçã", "jaca"];

echo "<h3>Ordem Crescente</h3>";

sort($frutas);

for($i = 0; $i < count($frutas); $i++){
    echo $frutas[$i] . "<br>";
}

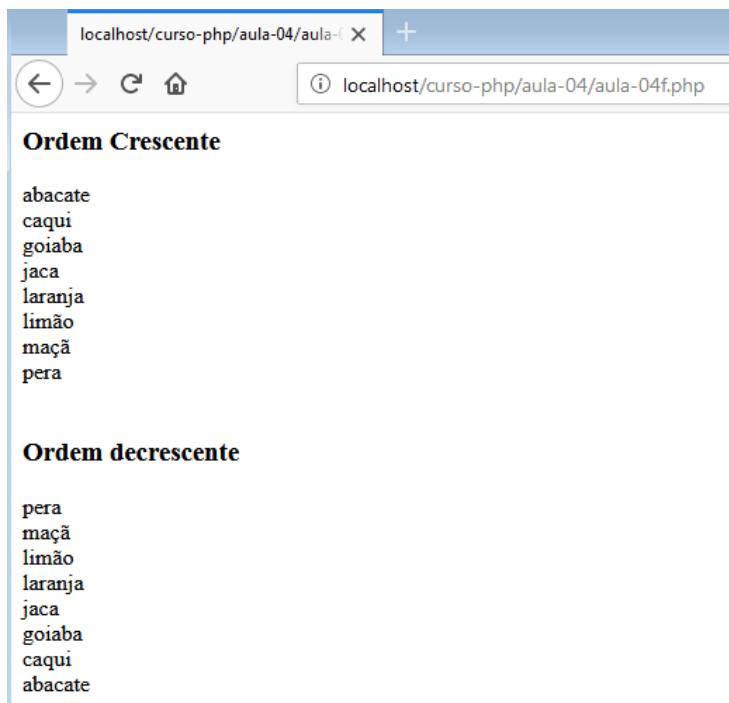
echo "<br>";

echo "<h3>Ordem decrescente</h3>";

rsort($frutas);

for($i = 0; $i < count($frutas); $i++){
    echo $frutas[$i] . "<br>";
}

?>
```



## Ordenação de arrays associativos

### aula-04/aula-04g.php

```
<?php

$pessoas = ["Ricardo" => 53, "Lucas" => 31, "Paulo" => 43, "Mateus" => 25,
"Alexandre" => 36];

asort($pessoas);

echo "<h3>Ordem Crescente por valor</h3>";

var_dump($pessoas);

arsort($pessoas);

echo "<br>";

echo "<h3>Ordem Decrescente por valor</h3>";

var_dump($pessoas);

echo "<br>";

ksort($pessoas);

echo "<h3>Ordem Crescente por chave</h3>";

var_dump($pessoas);

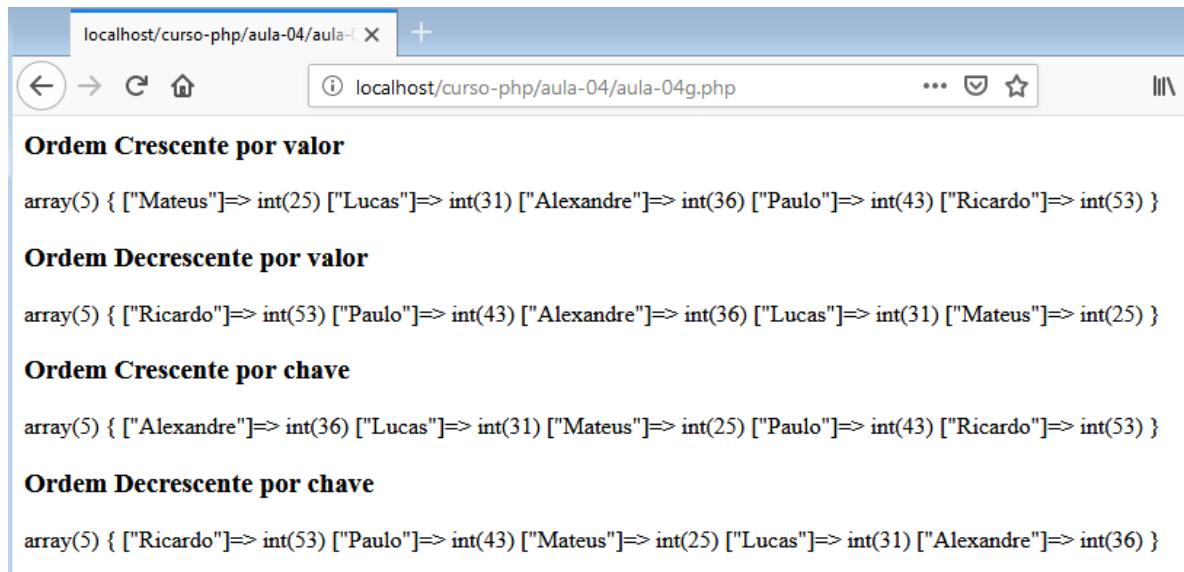
echo "<br>";

krsort($pessoas);

echo "<h3>Ordem Decrescente por chave</h3>";

var_dump($pessoas);

?>
```



A screenshot of a web browser window. The address bar shows the URL `localhost/curso-php/aula-04/aula-04g.php`. The page content displays the following text:

**Ordem Crescente por valor**

```
array(5) { ["Mateus"]=> int(25) ["Lucas"]=> int(31) ["Alexandre"]=> int(36) ["Paulo"]=> int(43) ["Ricardo"]=> int(53) }
```

**Ordem Decrescente por valor**

```
array(5) { ["Ricardo"]=> int(53) ["Paulo"]=> int(43) ["Alexandre"]=> int(36) ["Lucas"]=> int(31) ["Mateus"]=> int(25) }
```

**Ordem Crescente por chave**

```
array(5) { ["Alexandre"]=> int(36) ["Lucas"]=> int(31) ["Mateus"]=> int(25) ["Paulo"]=> int(43) ["Ricardo"]=> int(53) }
```

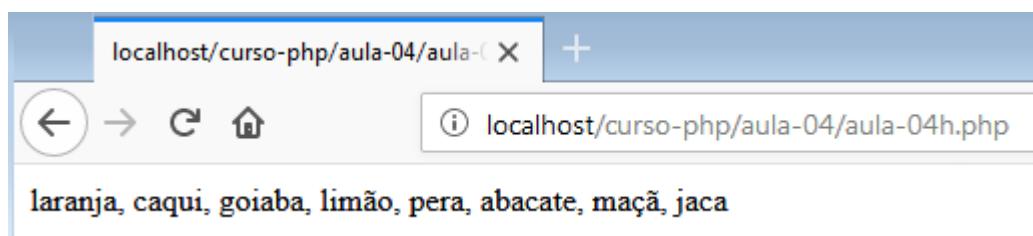
**Ordem Decrescente por chave**

```
array(5) { ["Ricardo"]=> int(53) ["Paulo"]=> int(43) ["Mateus"]=> int(25) ["Lucas"]=> int(31) ["Alexandre"]=> int(36) }
```

## Array para string (implode)

### aula-04/aula-04h.php

```
<?php  
  
$frutas = ["laranja", "caqui", "goiaba", "limão", "pera", "abacate", "maçã", "jaca"];  
  
$texto = implode(", ", $frutas);  
  
echo $texto;  
  
?>
```



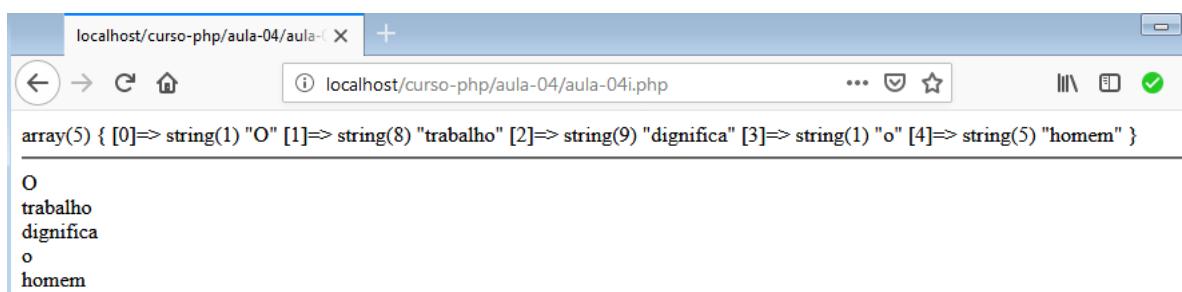
A screenshot of a web browser window. The address bar shows the URL `localhost/curso-php/aula-04/aula-04h.php`. The page content displays the following text:

laranja, caqui, goiaba, limão, pera, abacate, maçã, jaca

## String para Array (explode)

### aula-04/aula-04i.php

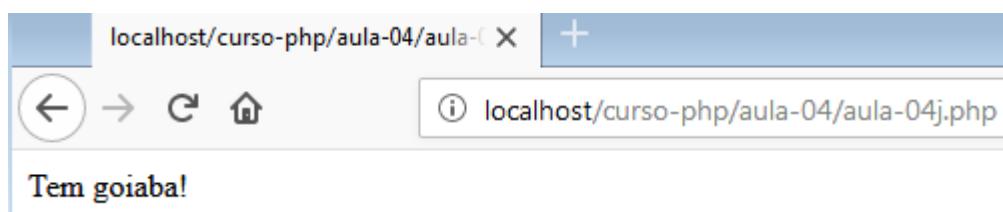
```
<?php  
  
$texto = "O trabalho significa o homem";  
  
$array = explode(" ", $texto);  
  
var_dump($array);  
  
echo "<br><hr>";  
  
foreach($array as $palavra){  
    echo $palavra . "<br>";  
}  
  
?>
```



## Checagem de valor no Array (in\_array)

### aula-04/aula-04j.php

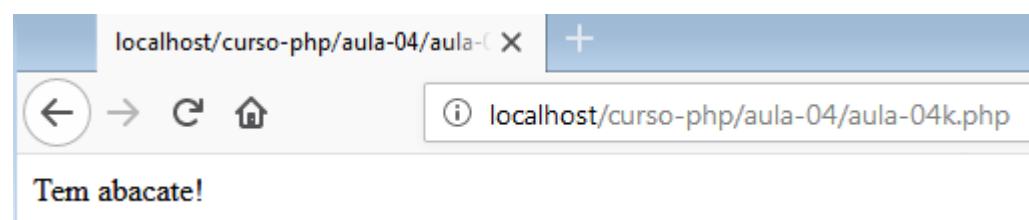
```
<?php  
  
$frutas = ["laranja", "caqui", "goiaba", "limão", "pera", "abacate", "maçã", "jaca"];  
  
if(in_array("goiaba", $frutas)){  
    echo "Tem goiaba!";  
} else {  
    echo "Não tem goiaba!";  
}  
  
?>
```



## Checagem de chave no Array (array\_key\_exists)

### aula-04/aula-04k.php

```
<?php  
  
$frutas = ["laranja" => 8.25, "goiaba" => 3.55, "limão" => 3.15, "abacate" =>  
4.75];  
  
if(array_key_exists("abacate", $frutas)){  
    echo "Tem abacate!";  
} else {  
    echo "Não tem abacate!";  
}  
  
?>
```



## Executar uma função em cada elemento (array\_map)

### aula-04/aula-04I.php

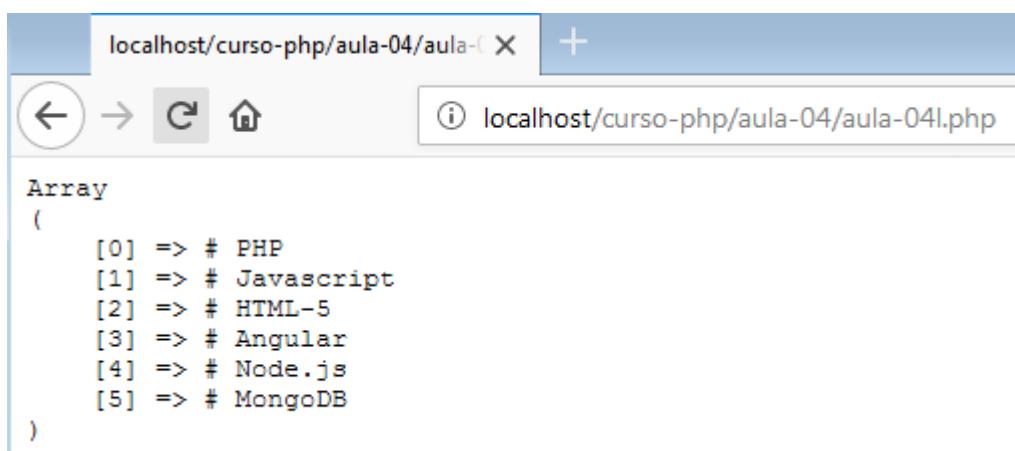
```
<?php

$conhecimentos = ["PHP", "Javascript", "HTML-5", "Angular", "Node.js",
"MongoDB"];

function hastag($x){
    return "# " . $x;
}

echo "<pre>";
print_r(array_map('hastag', $conhecimentos));
echo "</pre>";

?>
```



The screenshot shows a web browser window with the URL `localhost/curso-php/aula-04/aula-04I.php` in the address bar. The page content displays an array of hashtags generated by the `array_map` function:

```
Array
(
    [0] => # PHP
    [1] => # Javascript
    [2] => # HTML-5
    [3] => # Angular
    [4] => # Node.js
    [5] => # MongoDB
)
```

## Filtrar elementos do array por condição (array\_filter)

### aula-04/aula-04m.php

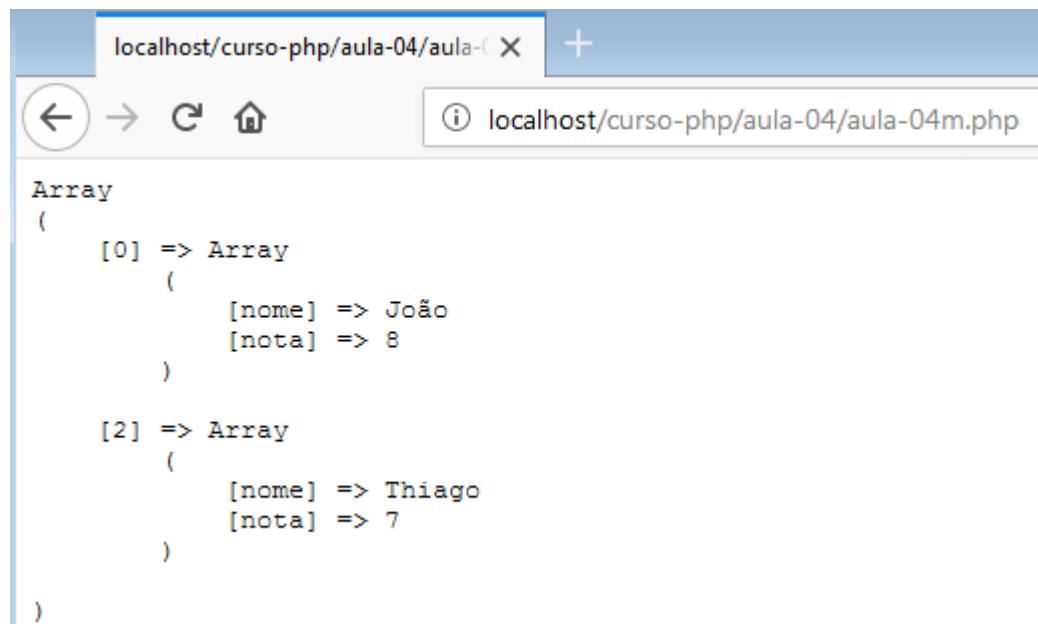
```
<?php

$alunos = [
    ["nome" => "João", "nota" => 8],
    ["nome" => "Maria", "nota" => 4],
    ["nome" => "Thiago", "nota" => 7]
];

$alunos_aprovados = array_filter($alunos, function($aluno){
    return $aluno['nota'] >= 7;
});

echo "<pre>";
print_r($alunos_aprovados);
echo "</pre>";

?>
```



```
Array
(
    [0] => Array
        (
            [nome] => João
            [nota] => 8
        )

    [2] => Array
        (
            [nome] => Thiago
            [nota] => 7
        )
)
```

## Variáveis para Array (compact)

### aula-04/aula-04n.php

```
<?php

$nome = "João";
$cpf = "123";
$rg = "8888";
$filhos = ["Maria", "Thiago", "Carlos"];

$result = compact("nome", "cpf", "rg", "filhos");

echo "<pre>";
print_r($result);
echo "</pre>";

?>
```

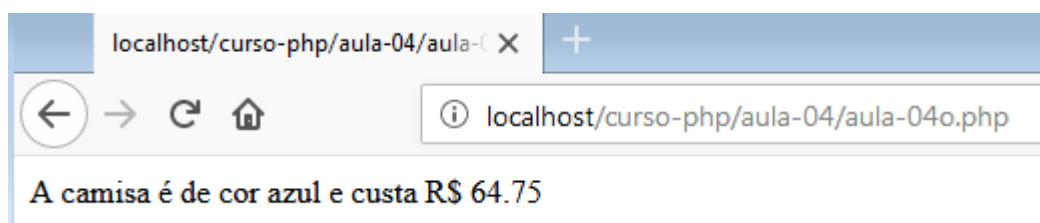


```
localhost/curso-php/aula-04/aula-04n.php +  
← → ⌂ ⌂ i localhost/curso-php/aula-04/aula-04n.php  
  
Array  
(  
    [nome] => João  
    [cpf] => 123  
    [rg] => 8888  
    [filhos] => Array  
        (  
            [0] => Maria  
            [1] => Thiago  
            [2] => Carlos  
        )  
)
```

## Array para variáveis (list)

### aula-04/aula-04o.php

```
<?php  
  
$info = ["camisa", "azul", 64.75];  
  
// Listando todas as variáveis  
  
list($produto, $cor, $preco) = $info;  
  
echo "A " . $produto . " é de cor " . $cor . " e custa R$ " . $preco;  
  
?>
```



## Array column (recuperar dados por key)

### aula-04/aula-04p.php

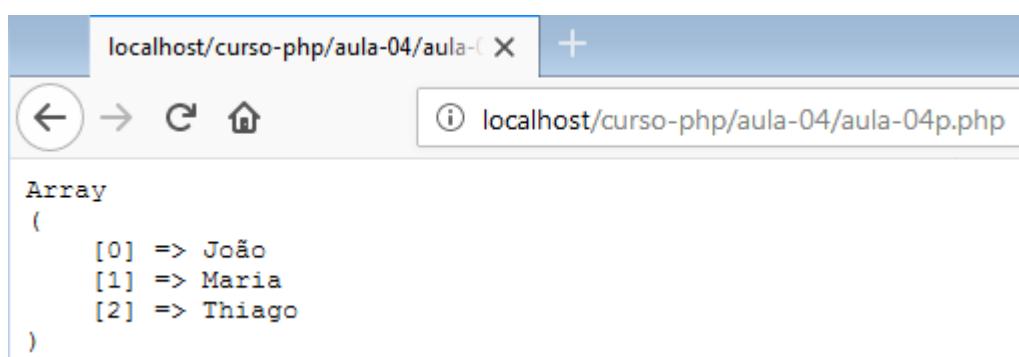
```
<?php

$pessoas = [
    ["id" => 1, "nome" => "João", "idade" => 43],
    ["id" => 2, "nome" => "Maria", "idade" => 16],
    ["id" => 3, "nome" => "Thiago", "idade" => 18]
];

$nomes = array_column($pessoas, 'nome');

echo "<pre>";
print_r($nomes);
echo "</pre>";

?>
```



```
Array
(
    [0] => João
    [1] => Maria
    [2] => Thiago
)
```

## Array column (recuperar dados por key com index)

### aula-04/aula-04q.php

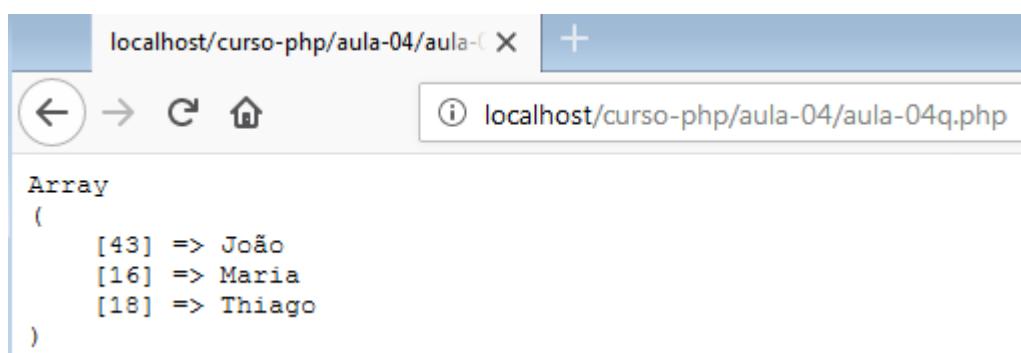
```
<?php

$pessoas = [
    ["id" => 1, "nome" => "João", "idade" => 43],
    ["id" => 2, "nome" => "Maria", "idade" => 16],
    ["id" => 3, "nome" => "Thiago", "idade" => 18]
];

$nomes = array_column($pessoas, 'nome', 'idade');

echo "<pre>";
print_r($nomes);
echo "</pre>";

?>
```



```
Array
(
    [43] => João
    [16] => Maria
    [18] => Thiago
)
```

## Array como pilha (array\_push, array\_pop, array\_shift)

### aula-04/aula-04r.php

```
<?php

$alunos = ["Carlos", "Maria"];

array_push($alunos, "Francisco", "Osvaldo");

echo "<pre>";
print_r($alunos);
echo "</pre>";

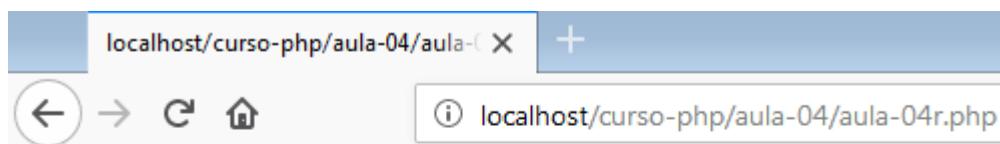
array_pop($alunos);

echo "<pre>";
print_r($alunos);
echo "</pre>";

array_shift($alunos);

echo "<pre>";
print_r($alunos);
echo "</pre>";

?>
```



```
Array
(
    [0] => Carlos
    [1] => Maria
    [2] => Francisco
    [3] => Osvaldo
)

Array
(
    [0] => Carlos
    [1] => Maria
    [2] => Francisco
)

Array
(
    [0] => Maria
    [1] => Francisco
)
```

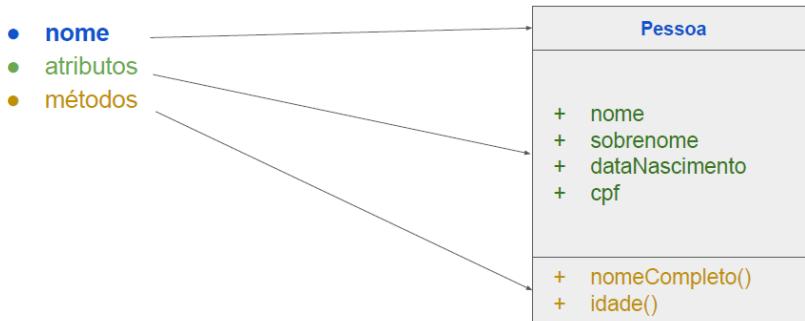
# Aula 05 - Classes, objetos, herança e visibilidade

## PHP OO

- A programação orientada a objetos (POO) é um paradigma baseado no conceito de "objetos", que podem conter dados, na forma de campos (atributos/propriedades) e instruções de código, na forma de funções (métodos).
- Consolidado no PHP 5, os recursos de Orientação a Objetos suportados pela linguagem estão a inclusão de visibilidade, classes e métodos abstratos e final, métodos mágicos, interfaces, clonagem e *type hint*.
- O PHP trata objetos da mesma maneira que referências ou manipuladores, onde cada variável contém uma referência a um objeto ao invés de uma cópia de todo o objeto.

## Classes e Objetos

### Classe



### Instanciando - new()

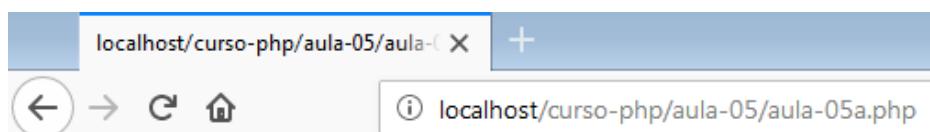
- new() cria uma referência a um objeto.

### Object Operator (->)

- Para acessar ou modificar valores ou métodos usa-se o Object Operator (arrow) "`->`"

## **aula-05/aula-05a.php**

```
<?php  
  
class Pessoa {  
  
    public $nome;  
    public $sobrenome;  
  
    public function nomeCompleto(){  
        echo "Nome: " . $this->nome." ".$this->sobrenome . "<br>";  
    }  
  
}  
  
$pessoa1 = new Pessoa(); // É uma referência a um objeto de Pessoa  
$pessoa1->nome = "Carlos";  
$pessoa1->sobrenome = "Silva";  
$pessoa1->nomeCompleto();  
  
$pessoa2 = new Pessoa();  
$pessoa2->nome = "Regina";  
$pessoa2->sobrenome = "Santos";  
$pessoa2->nomeCompleto();
```



Nome: Carlos Silva  
Nome: Regina Santos

## Classes e objetos

```
1 <?php
2
3 v class Pessoa {
4
5     public $nome;
6     public $sobrenome;
7
8 v     public function nomeCompleto(){
9         echo "Nome: " . $this->nome." ".$this->sobrenome . PHP_EOL;
10    }
11
12 }
13
14 $pessoal = new Pessoa();
15 $pessoal->nome = "Carlos";
16 $pessoal->sobrenome = "Silva";
17
18 var_dump($pessoal);
19
20 $pessoal->nomeCompleto();
21
22 $pessoal2 = new Pessoa();
23 $pessoal2->nome = "Regina";
24 $pessoal2->sobrenome = "Santos";
25
26 var_dump($pessoal2);
27
28 $pessoal2->nomeCompleto();
```

Output

Performance

VLD opcodes

References

Branches

Output for 7.3.0 - 7.3.27, 7.4.0 - 7.4.16, 8.0.0 - 8.0.3

```
object(Pessoa)#1 (2) {
    ["nome"]=>
        string(6) "Carlos"
    ["sobrenome"]=>
        string(5) "Silva"
}
Nome: Carlos Silva
object(Pessoa)#2 (2) {
    ["nome"]=>
        string(6) "Regina"
    ["sobrenome"]=>
        string(6) "Santos"
}
Nome: Regina Santos
```

## clone

- Para ter uma nova pessoa é necessário usar **new**, a não ser que se use um recurso chamado **clone** que criará uma nova pessoa com os mesmos atributos da pessoa clonada. Esses atributos posteriormente podem ou não serem modificados.

Exemplo:

```
$pessoa3 = clone $pessoa1;
```

## Classes e objetos

```
1 <?php
2
3 * class Pessoa {
4
5     public $nome;
6     public $sobrenome;
7
8 *     public function nomeCompleto(){
9         echo "Nome: " . $this->nome." ".$this->sobrenome . PHP_EOL;
10    }
11 }
12 }
13
14 $pessoa1 = new Pessoa();
15 $pessoa1->nome = "Carlos";
16 $pessoa1->sobrenome = "Silva";
17
18 var_dump($pessoa1);
19
20 $pessoa1->nomeCompleto();
21
22 $pessoa2 = new Pessoa();
23 $pessoa2->nome = "Regina";
24 $pessoa2->sobrenome = "Santos";
25
26 var_dump($pessoa2);
27
28 $pessoa2->nomeCompleto();
29
30 $pessoa3 = clone $pessoa1;
31 $pessoa3->nome = "José";
32
33 var_dump($pessoa1);
34 var_dump($pessoa3);
```

| Output   | Performance | VLD opcodes | References | Branches |
|--|-------------|-------------|------------|----------|
| <p>Output for 7.3.0 - 7.3.27, 7.4.0 - 7.4.16, 8.0.0 - 8.0.3</p> <pre>object(Pessoa)#1 (2) {     ["nome"]=&gt;         string(6) "Carlos"     ["sobrenome"]=&gt;         string(5) "Silva" } Nome: Carlos Silva object(Pessoa)#2 (2) {     ["nome"]=&gt;         string(6) "Regina"     ["sobrenome"]=&gt;         string(6) "Santos" } Nome: Regina Santos object(Pessoa)#1 (2) {     ["nome"]=&gt;         string(6) "Carlos"     ["sobrenome"]=&gt;         string(5) "silva" } object(Pessoa)#3 (2) {     ["nome"]=&gt;         string(5) "José"     ["sobrenome"]=&gt;         string(5) "Silva" }</pre> |             |             |            |          |

## Referência ao objeto (\$this)

- \$this permite fazer uma referência dinâmica ao objeto corrente.

Exemplo:

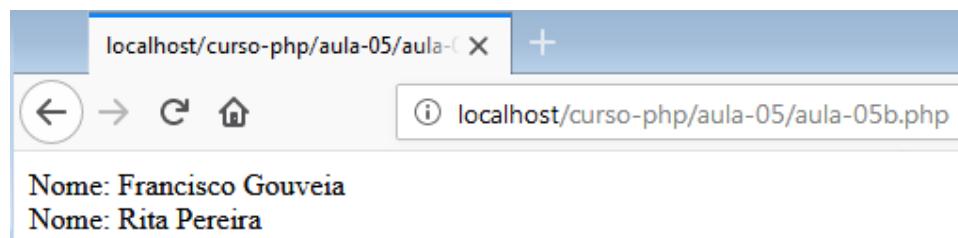
```
class Pessoa {  
  
    public $nome;  
    public $sobrenome;  
  
    public function nomeCompleto(){  
        echo "Nome: " . $this->nome . " " . $this->sobrenome . "<br>";  
    }  
  
}
```

## Método Construtor

É um método (função) especial que é executado toda vez que um objeto é instanciado. O método construtor permite inicializar variáveis e não é executado no clone.

### aula-05/aula-05b.php

```
<?php  
  
class Pessoa {  
  
    public $nome;  
    public $sobrenome;  
  
    public function __construct($nome, $sobrenome){  
        $this->nome = $nome;  
        $this->sobrenome = $sobrenome;  
    }  
  
    public function nomeCompleto(){  
        echo "Nome: " . $this->nome . ".$this->sobrenome . "<br>";  
    }  
  
}  
  
$pessoa1 = new Pessoa("Francisco", "Gouveia");  
$pessoa1->nomeCompleto();  
  
$pessoa2 = new Pessoa("Rita", "Pereira");  
$pessoa2->nomeCompleto();
```



## Método Destruitor

- É executado quando o objeto morre (quando o código acaba), ou quando um objeto não tem mais referência nenhuma.

### aula-05/aula-05c.php

```
<?php

class Pessoa {

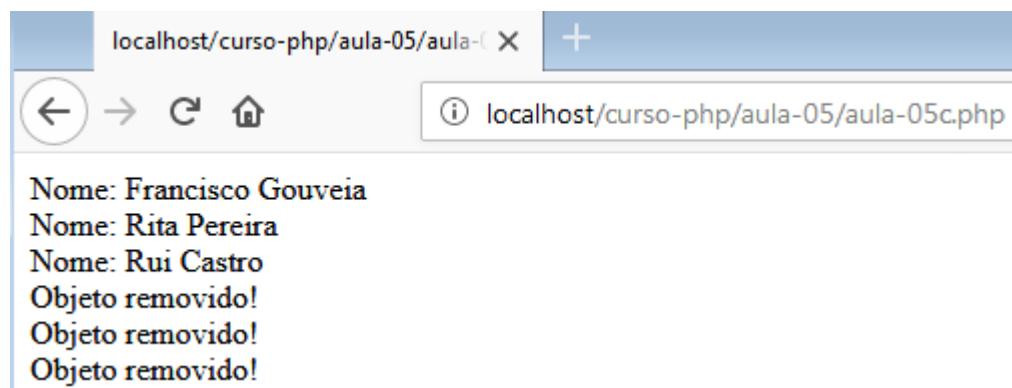
    public $nome;
    public $sobrenome;

    public function __construct($nome, $sobrenome){
        $this->nome = $nome;
        $this->sobrenome = $sobrenome;
    }

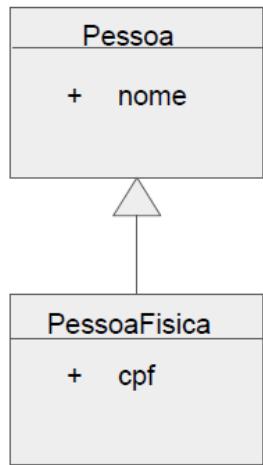
    public function nomeCompleto(){
        echo "Nome: " . $this->nome." ".$this->sobrenome . "<br>";
    }

    public function __destruct(){
        echo "Objeto removido! <br>";
    }
}

$pessoa1 = new Pessoa("Francisco", "Gouveia");
$pessoa1->nomeCompleto();
$pessoa2 = new Pessoa("Rita", "Pereira");
$pessoa2->nomeCompleto();
$pessoa3 = new Pessoa("Rui", "Castro");
$pessoa3->nomeCompleto();
```



## Herança (extends)



### aula-05/aula-05d.php

```
<?php

class Pessoa {
    public $nome;

    public function exibirNome(){
        echo "<p>Nome: " . $this->nome . "</p>";
    }
}

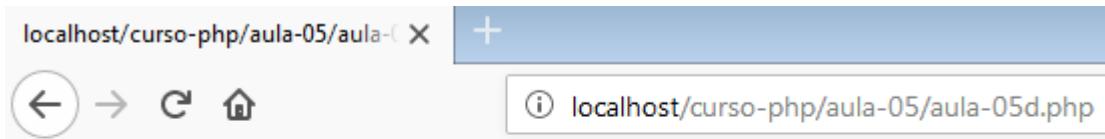
class PessoaFisica extends Pessoa {
    public $cpf;
}

class PessoaJuridica extends Pessoa {
    public $cnpj;
}

$pf1 = new PessoaFisica();
$pf1->nome = "Gilberto";
$pf1->cpf = "123";
$pf1->exibirNome();

$pj1 = new PessoaJuridica();
$pj1->nome = "Lucas & Associados Ltda.";
$pj1->cnpj = "4567";
$pj1->exibirNome();

?>
```



Nome: Gilberto

Nome: Lucas & Associados Ltda.

- O PHP não possui suporte a herança múltipla.

## Method overriding e parent

A substituição de métodos (method overriding), é um recurso que permite que uma subclasse ou classe filha forneça uma implementação específica de um método que já é fornecido por uma de suas superclasses ou classes pai.

A implementação na subclasse substitui (override) a implementação da superclasse, fornecendo um método que possui o mesmo nome, mesmos parâmetros ou assinatura, e o mesmo tipo de retorno que o método na classe pai.

### aula-05/aula-05e.php

```
<?php

class Pessoa {
    public $nome;

    public function exibirNome(){
        echo "<p>Nome: " . $this->nome . "</p>";
    }
}

class PessoaFisica extends Pessoa {
    public $cpf;

    public function exibirNome(){
        echo "Nome: " . $this->nome . " - CPF: " . $this->cpf . " <br>";
    }
}
```

```

class PessoaJuridica extends Pessoa {
    public $cnpj;

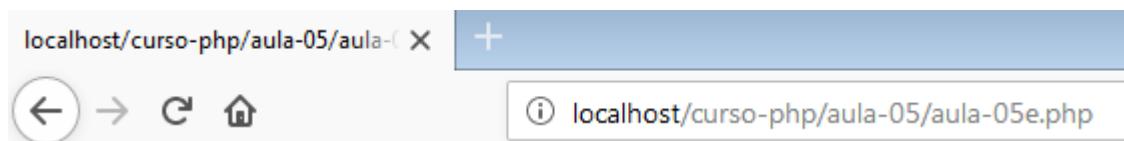
    public function exibirNome(){
        echo "Nome: " . $this->nome . " - CNPJ: " . $this->cnpj . " <br>";
    }
}

$pf1 = new PessoaFisica();
$pf1->nome = "Gilberto";
$pf1->cpf = "123";
$pf1->exibirNome();

$pj1 = new PessoaJuridica();
$pj1->nome = "Lucas & Associados Ltda.";
$pj1->cnpj = "4567";
$pj1->exibirNome();

?>

```



## Parent

Apesar do recurso de overriding, ainda é possível que a classe filha "execute" um método da classe pai através do recurso parent.

### **aula-05/aula-05f.php**

```

<?php

class Pessoa {
    public $nome;

    public function exibirNome(){
        echo "Nome: " . $this->nome;
    }
}

```

```

class PessoaFisica extends Pessoa {
    public $cpf;

    public function exibirNome(){
        echo parent::exibirNome() . " - CPF: " . $this->cpf . " <br>";
    }
}

class PessoaJuridica extends Pessoa {
    public $cnpj;

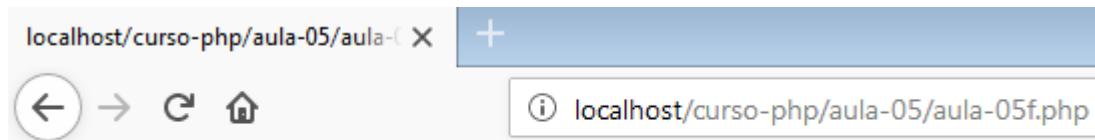
    public function exibirNome(){
        echo parent::exibirNome() . " - CNPJ: " . $this->cnpj . " <br>";
    }
}

$pf1 = new PessoaFisica();
$pf1->nome = "Gilberto";
$pf1->cpf = "123";
$pf1->exibirNome();

$pj1 = new PessoaJuridica();
$pj1->nome = "Lucas & Associados Ltda.";
$pj1->cnpj = "4567";
$pj1->exibirNome();

?>

```



Nome: Gilberto - CPF: 123  
 Nome: Lucas & Associados Ltda. - CNPJ: 4567

## Visibilidade

PHP OO (visibilidade)

**public:** torna uma variável/método disponível de qualquer lugar, outras classes e instâncias do objeto.

- Mesma classe que foi declarada
- Classes que herdam
- As classes que herdam a classe declarada acima.
- Quaisquer elementos estrangeiros fora dessa classe também podem acessar estes elementos

**protected:** escopo quando você quiser fazer a sua função ou variável visível em todas as classes que estendem a classe atual, incluindo a classe pai.

Quando você declara um método (função) ou uma propriedade (atributo) como protegido, esses métodos e propriedades podem ser acessados por a mesma classe que declarou isso.

As classes que herdam a classe declarada acima. Os membros de outsiders não podem acessar essas variáveis. "Outsiders" no sentido de que eles não são instâncias de objetos da própria classe declarada.

**private:** escopo quando você quer que a sua função ou variável seja visível apenas em sua própria classe.

Quando você declara um método (função) ou uma propriedade (variável) como privada, esses métodos e propriedades podem ser acessados por a mesma classe que declarou isso.

## **aula-05/aula-05g.php**

```
<?php

class Pessoa {
    public $texto1;
    private $texto2;
    protected $texto3;

    public function setPrivado($texto2){
        $this->texto2 = $texto2;
    }

    public function getPrivado(){
        return $this->texto2;
    }

    public function setProtegido($texto3){
        $this->texto3 = $texto3;
    }

    public function getProtegido(){
        return $this->texto3;
    }
}

class PessoaFisica extends Pessoa {

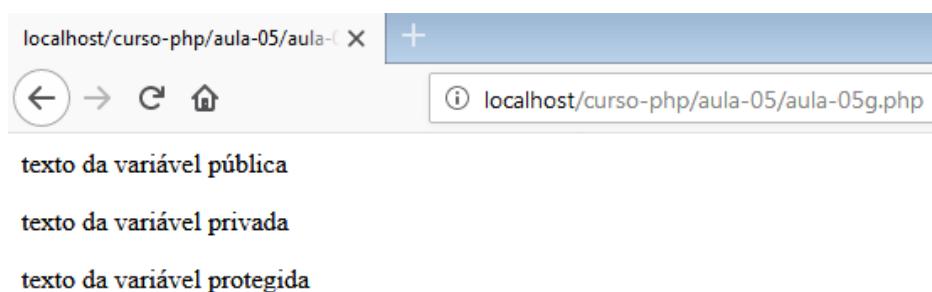
}

$pessoa = new Pessoa();
$pessoa->texto1 = "texto da variável pública";
$pessoa->setPrivado("texto da variável privada");
$pessoa->setProtegido("texto da variável protegida");

echo "<p>" . $pessoa->texto1 . "</p>";
echo "<p>" . $pessoa->getPrivado() . "</p>";

$pf = new PessoaFisica();
echo "<p>" . $pessoa->getProtegido() . "</p>";

?>
```



## **aula-05/aula-05h.php**

```
<?php

class Pessoa {
    public $texto1;
    private $texto2;
    protected $texto3;

    public function setPrivado($texto2){
        $this->texto2 = $texto2;
    }

    public function getPrivado(){
        return $this->texto2;
    }

    public function setProtegido($texto3){
        $this->texto3 = $texto3;
    }

    public function getProtegido(){
        return $this->texto3;
    }

    private function segredo(){
        return "<p>Segredo revelado!</p>";
    }

    public function revelarSegredo(){
        return $this->segredo();
    }
}

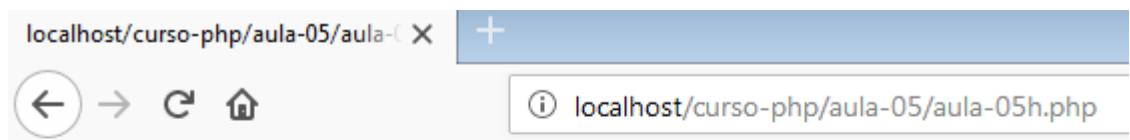
class PessoaFisica extends Pessoa {

    $pessoa = new Pessoa();
    $pessoa->texto1 = "texto da variável pública";
    $pessoa->setPrivado("texto da variável privada");
    $pessoa->setProtegido("texto da variável protegida");

    echo "<p>" . $pessoa->texto1 . "</p>";
    echo "<p>" . $pessoa->getPrivado() . "</p>";
    echo $pessoa->revelarSegredo();

    $pf = new PessoaFisica();
    echo "<p>" . $pessoa->getProtegido() . "</p>";

?>
```



texto da variável pública

texto da variável privada

Segredo revelado!

texto da variável protegida

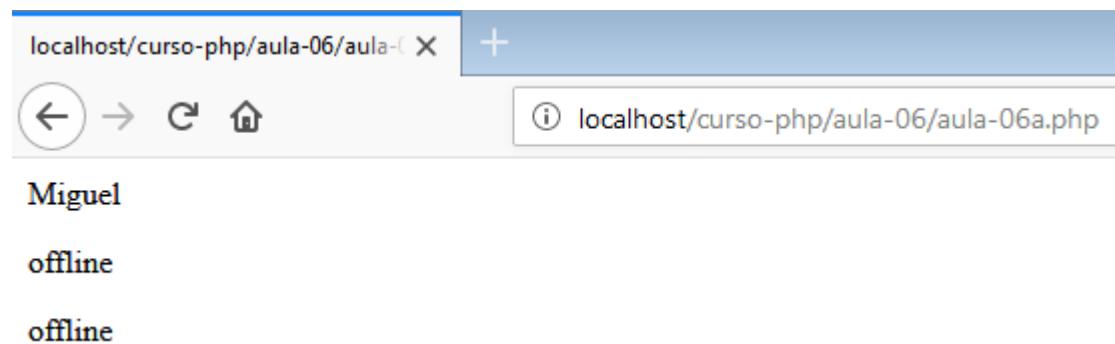
# Aula 06 - Atributos e Métodos Estáticos, Trait, Interface, Classe Abstrata

## Atributos estáticos

Na programação orientada a objetos, atributo estático é conhecido como "**variável de classe**". Um atributo estático de uma determinada classe tem seus valores compartilhados em todas as instâncias (objetos) ou diretamente na classe utilizando o operador `::`

### aula-06/aula-06a.php

```
<?php  
  
class Teste{  
  
    public static $status = 'online';  
    public $nome;  
  
}  
  
$t1 = new Teste();  
$t1->nome="Miguel";  
Teste::$status='offline';  
  
echo "<p>" . $t1->nome . "</p>";  
echo "<p>" . Teste::$status . "</p>";  
echo "<p>" . $t1::$status . "</p>";  
  
?>
```



## Métodos estáticos

De maneira similar, um método estático é "procedimento da classe". Um método estático de uma determinada classe pode ser acessado em todas as instâncias (objetos) ou diretamente na classe utilizando o operador `::`:

É válido salientar que é proibido utilizar o `$this` dentro (no contexto) de um método estático.

### aula-06/aula-06b.php

```
<?php

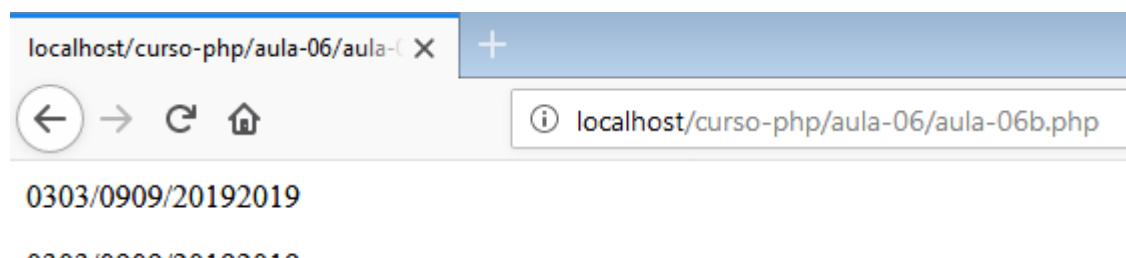
class Util{

    public static function hoje(){
        return date('dd/mm/YY');
    }

}

$util = new Util();
echo "<p>" . $util->hoje() . "</p>";
echo "<p>" . Util::hoje() . "</p>";

?>
```



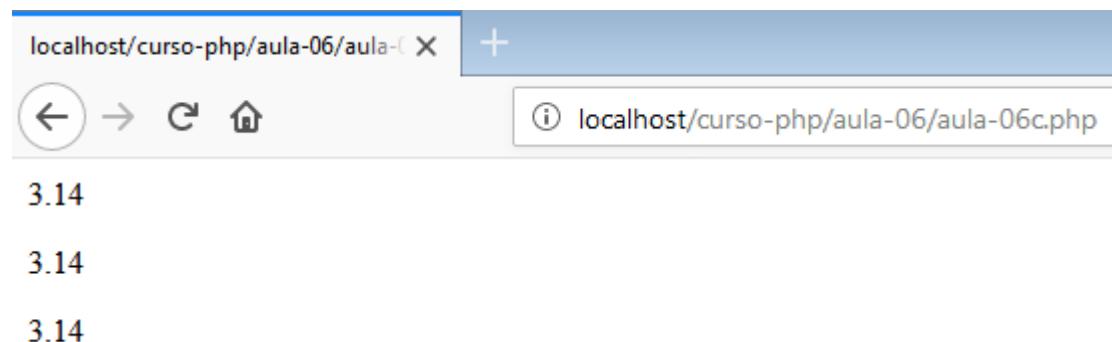
## Constantes de classe

Similar as constantes usando o `define()`, o valor de uma constante nunca pode ser alterado.

O acesso é similar aos atributos estáticos, utilizando o `::` porém sem o `$`.

### aula-06/aula-06c.php

```
<?php  
  
class Matematica{  
  
    const PI = 3.14;  
  
    function valorDePi(){  
        echo self::PI;  
    }  
  
}  
  
echo "<p>" . Matematica::PI . "</p>";  
echo "<p>" . Matematica::valorDePi() . "</p>";  
$mat = new Matematica();  
echo "<p>" . $mat->valorDePi() . "</p>";  
  
?>
```



## Constantes de classe (visibilidade)

Válido somente à partir do PHP 7.1

### aula-06/aula-06d.php

```
<?php

class Matematica{

    private const PI = 3.14;

    function valorDePi(){
        echo self::PI;
    }

}

echo "<p>" . Matematica::PI . "</p>";
echo "<p>" . Matematica::valorDePi() . "</p>";
$mat = new Matematica();
echo "<p>" . $mat->valorDePi() . "</p>";

?>
```



## Final Method

Não permite sobrescrita.

O PHP 5 introduziu a palavra-chave final, que previne que classes filhas sobrescrevam um método que esteja prefixado em sua definição com a palavra final.

### aula-06/aula-06e.php

```
<?php  
  
class Pai{  
  
    final public function fazAlgo(){  
        echo "Oi";  
    }  
  
}  
  
class Filho extends Pai{  
  
    final public function fazAlgo(){  
        echo "Olá";  
    }  
  
}  
  
$filho = new Filho();  
echo "<p>" . $filho->fazAlgo() . "</p>";  
  
?>
```

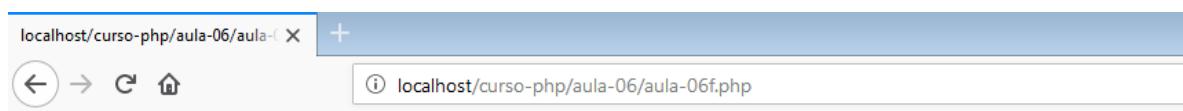


**Fatal error:** Cannot override final method Pai::fazAlgo() in C:\xampp\htdocs\curso-php\aula-06\aula-06e.php on line 17

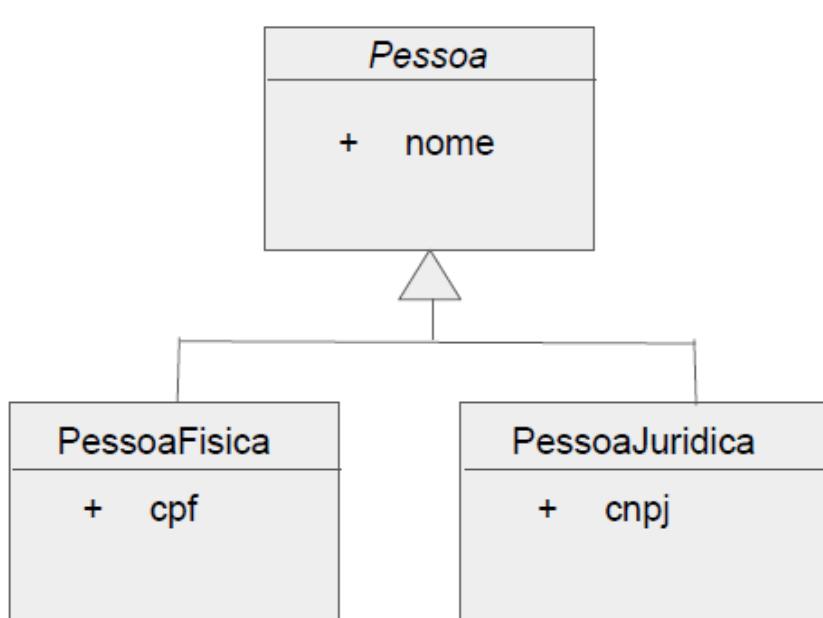
Se a própria classe estiver definida como final, ela não pode ser estendida.

## **aula-06/aula-06f.php**

```
<?php  
  
final class Pai{  
  
    public function fazAlgo(){  
        echo "Oi";  
    }  
  
}  
  
class Filho extends Pai{  
  
}  
  
$filho = new Filho();  
echo "<p>" . $filho->fazAlgo() . "</p>";  
  
?>
```



## **Classe abstrata**



Uma classe abstrata não pode ser instanciada.

## aula-06/aula-06g.php

```
<?php

abstract class Pessoa{
    public $nome;
}

class PessoaFisica extends Pessoa{
    public $cpf;
}

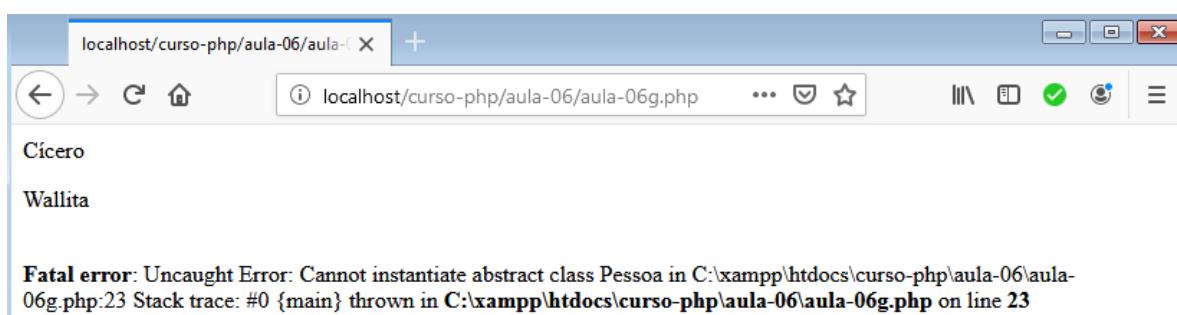
class PessoaJuridica extends Pessoa{
    public $cnpj;
}

$pf = new PessoaFisica();
$pf->nome = "Cícero";
echo "<p>" . $pf->nome . "</p>";

$pj = new PessoaJuridica();
$pj->nome = "Wallita";
echo "<p>" . $pj->nome . "</p>";

$pessoa = new Pessoa();
$pessoa->nome = "Alberto";
echo "<p>" . $pessoa->nome . "</p>",

?>
```



## Métodos abstratos

Métodos abstratos precisam ser implementados nas classes filhas.

### aula-06/aula-06h.php

```
<?php

abstract class Pessoa{
    public $nome;
    abstract public function exibirNumeroDocumento();
}

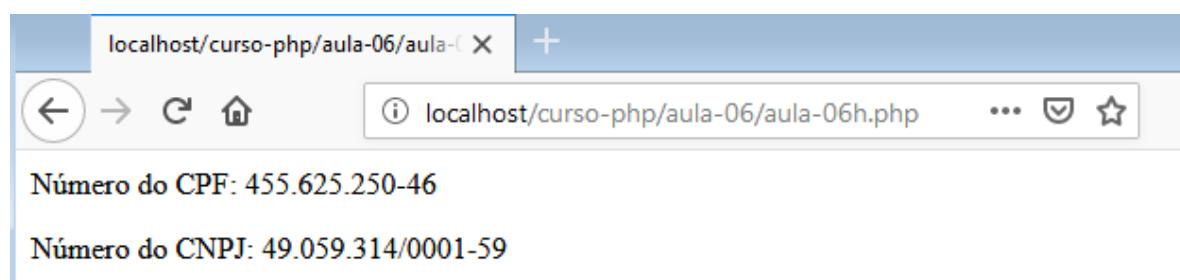
class PessoaFisica extends Pessoa{
    public $cpf;
    public function exibirNumeroDocumento(){
        echo "<p>Número do CPF: " . $this->cpf . "</p>";
    }
}

class PessoaJuridica extends Pessoa{
    public $cnpj;
    public function exibirNumeroDocumento(){
        echo "<p>Número do CNPJ: " . $this->cnpj . "</p>";
    }
}

$pf = new PessoaFisica();
$pf->cpf = '455.625.250-46';
$pf->exibirNumeroDocumento();

$pj = new PessoaJuridica();
$pj->cnpj = '49.059.314/0001-59';
$pj->exibirNumeroDocumento();

?>
```



# Interface

## Interface

- Interfaces permitem a criação de códigos que especificam quais métodos uma classe deve implementar, sem definir como esses métodos serão tratados.
- Interfaces permitem também que classes de diferentes hierarquias possam ter comportamentos similares.
- Interfaces são definidas da mesma forma que classes, mas com a palavra-chave **interface** substituindo **class** e com nenhum dos métodos tendo seu conteúdo definido.
- Todos os métodos declarados em uma interface devem ser públicos, essa é a natureza de uma interface.
- Interfaces PHP suportam herança múltipla e constantes

É obrigatório ter declarado o(s) método(s) de uma interface em todas as classes que a implementam.

## aula-06/aula-06i.php

```
<?php

interface MinhaInterface {
    public function fazAlgo();
    public function fazOutraCoisa();
}

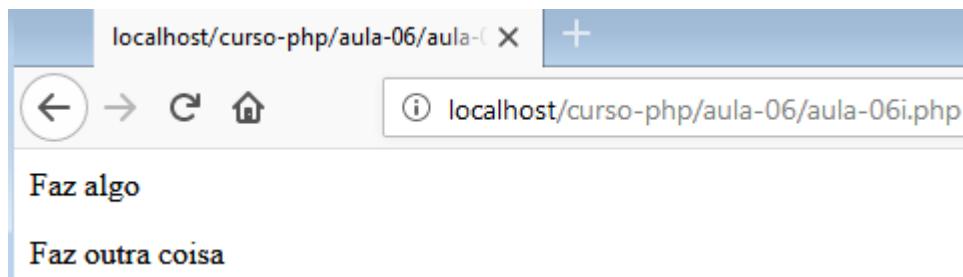
class Teste implements MinhaInterface{
    // erro: é obrigatório implementar o método Formatador
    // obs.: exceção se a classe Html for abstrata

    public function fazAlgo(){
        echo "<p>Faz algo</p>";
    }

    public function fazOutraCoisa(){
        echo "<p>Faz outra coisa</p>";
    }
}

$teste = new Teste();
$teste->fazAlgo();
$teste->fazOutraCoisa();

?>
```



## Interface (herança múltipla)

É obrigatório ter declarado o(s) método(s) de uma interface em todas as classes que a implementam.

### aula-06/aula-06j.php

```
<?php

interface Interface1 {
    public function fazAlgo();
}

interface Interface2 {
    public function fazOutraCoisa();
}

interface Interface3 extends Interface1, Interface2 {}

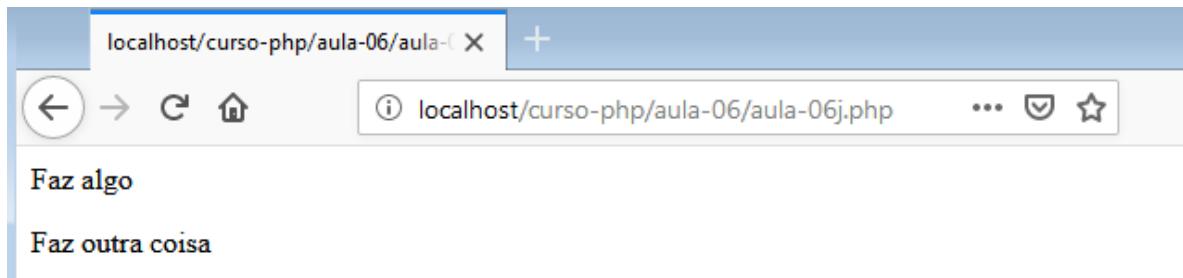
class Teste implements Interface3{
    // erro: é obrigatório implementar o método Formatador
    // obs.: exceção se a classe Html for abstrata

    public function fazAlgo(){
        echo "<p>Faz algo</p>";
    }

    public function fazOutraCoisa(){
        echo "<p>Faz outra coisa</p>";
    }
}

$teste = new Teste();
$teste->fazAlgo();
$teste->fazOutraCoisa();

?>
```



## Interface (usando typehint)

Neste exemplo, classes de diferentes taxonomias implementam uma mesma interface.

### aula-06/aula-06k.php

```
<?php

Interface sepultavel{
    public function CalcularValorDoCaixao(float $valor) : float;
}

class AnimalDomestico{}

class Felino extends AnimalDomestico implements sepultavel{
    public $peso;
    public function CalcularValorDoCaixao(float $valor) : float {
        return $valor * ($this->peso * 0.3);
    }
}

class HomoSapiens{}

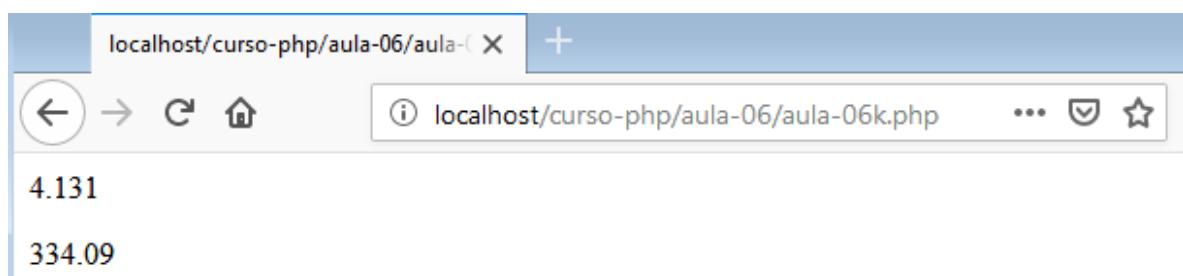
class Pessoa extends HomoSapiens implements sepultavel{
    public $altura;
    public $peso;
    public $circunferenciaAbdominal;
    public function CalcularValorDoCaixao(float $valor) : float {
        return $valor * ($this->altura * $this->peso) / 2 + $this-
>circunferenciaAbdominal;
    }
}

class Sepultamento{
    public $cotacao = 4.05;
    public function enterrar(Sepultavel $sepultavel) : float{
        return $sepultavel->CalcularValorDoCaixao($this->cotacao);
    }
}
```

```
$gato = new Felino();
$gato->peso = 3.4;
$s = new Sepultamento();
echo "<p>" . $s->enterrar($gato) . "</p>";

$p1 = new Pessoa();
$p1->altura = 1.70;
$p1->peso = 68;
$p1->circunferenciaAbdominal = 100;
$s2 = new Sepultamento();
echo "<p>" . $s2->enterrar($p1) . "</p>";

?>
```



## Trait

- Traits são mecanismos para reutilização de código em linguagens de heranças únicas, como PHP.
- Uma Trait destina-se a reduzir algumas limitações de herança única permitindo que um desenvolvedor reutilize conjuntos de métodos livremente em várias classes independentes que podem viver em diferentes taxonomias de classes.
- Não é possível instanciar um Trait por conta própria.
- É uma adição à herança tradicional e permite a composição horizontal do comportamento, isto é, a aplicação de membros de classe sem exigir herança.

### **aula-06/aula-06I.php**

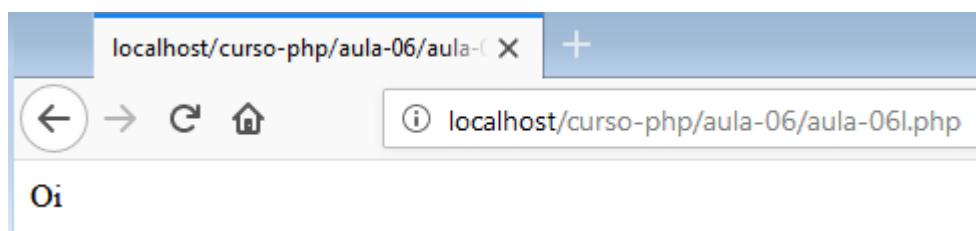
```
<?php

trait MinhaTrait{
    public function oi(){
        echo "<p>Oi</p>";
    }
}

class Teste {
    use MinhaTrait;
}

$teste = new Teste();
$teste->oi();

?>
```



## Trait (múltiplo uso)

### aula-06/aula-06m.php

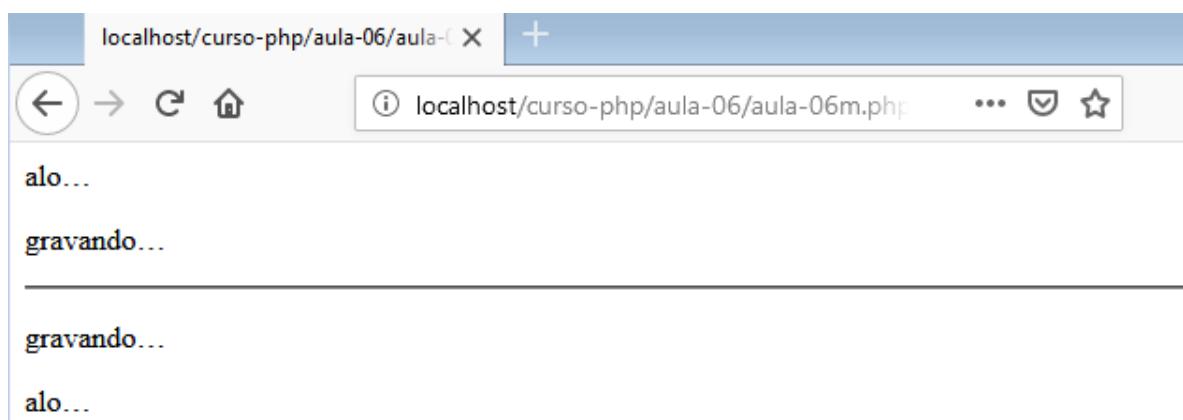
```
<?php

trait Telefone {
    public function ligar() {
        echo "<p>alo...</p>";
    }
}

trait Camera {
    public function filmar() {
        echo "<p>gravando...</p>";
    }
}

class Smartphone {
    use Telefone, Camera;
    public function videoChamada() {
        echo "<hr>";
        echo $this->filmar() . $this->ligar();
    }
}

$o = new Smartphone();
$o->ligar();
$o->filmar();
$o->videoChamada();
```



## Trait (precedência/instead of)

### aula-06/aula-06n.php

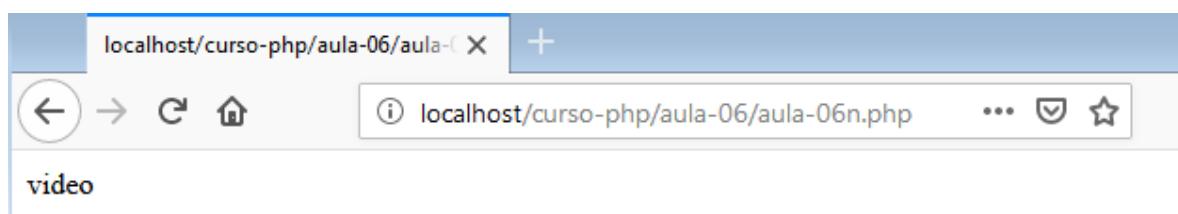
```
<?php

trait Video {
    public function tag() {
        echo 'video';
    }
}

trait Audio {
    public function tag() {
        echo 'audio';
    }
}

class Media {
    use Audio, Video {
        Video::tag insteadof Audio;
    } //sem esta resolução ocorrerá um Fatal Error.
}

$media = new Media();
echo $media->tag(); //resultado: <video>
```



## Trait (alias "as")

### aula-06/aula-06o.php

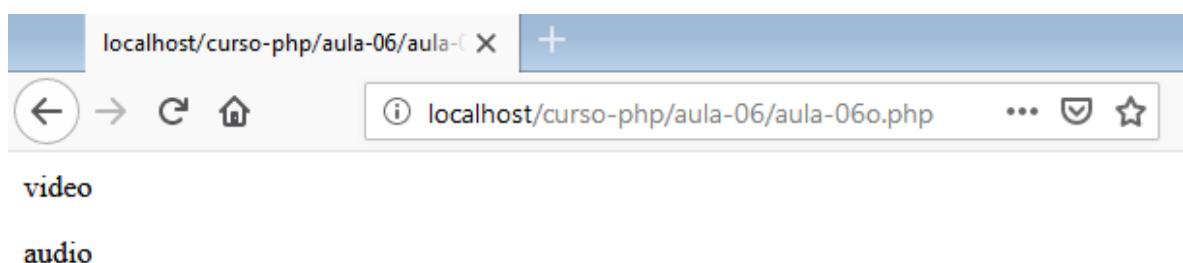
```
<?php

trait Video {
    public function tag() {
        echo "<p>video</p>";
    }
}

trait Audio {
    public function tag() {
        echo "<p>audio</p>";
    }
}

class Media {
    use Audio, Video {
        Video::tag insteadof Audio;
        Audio::tag as tagAudio;
    }
}

$media = new Media();
echo $media->tag();
echo $media->tagAudio();
```

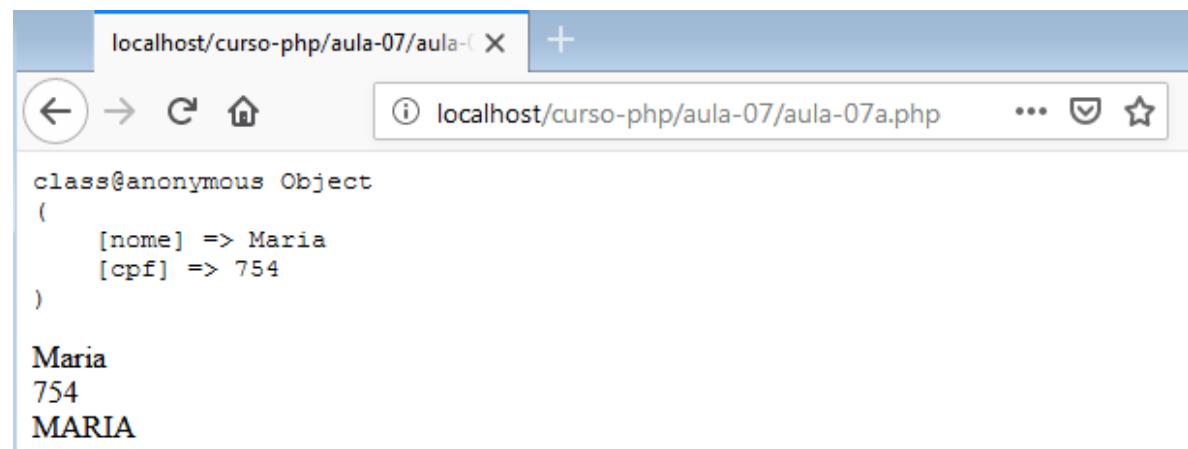


# Aula 07 - Classe anônima, associações, composição e agregação

## Classe anônima

**aula-07/aula-07a.php**

```
<?php  
  
$pessoa = new class {  
  
    public $nome = 'Maria';  
    public $cpf = '754';  
  
    public function getNome(){  
        return strtoupper($this->nome);  
    }  
};  
  
echo "<pre>";  
print_r($pessoa);  
echo "</pre>";  
  
echo $pessoa->nome . "<br>";  
echo $pessoa->cpf . "<br>";  
echo $pessoa->getNome();
```



The screenshot shows a web browser window with the following details:

- Address Bar:** localhost/curso-php/aula-07/aula-07a.php
- Toolbar:** Back, Forward, Stop, Home, Refresh, Address Bar, More, Star.
- Content Area:** Displays the output of a PHP script. It shows the definition of an anonymous class with properties \$nome and \$cpf, and a method getNome(). It then prints\_r()s the object, showing an array with keys [nome] and [cpf]. Finally, it prints the values of nome, cpf, and the result of getNome().

```
class@anonymous Object
(
    [nome] => Maria
    [cpf] => 754
)
Maria
754
MARIA
```

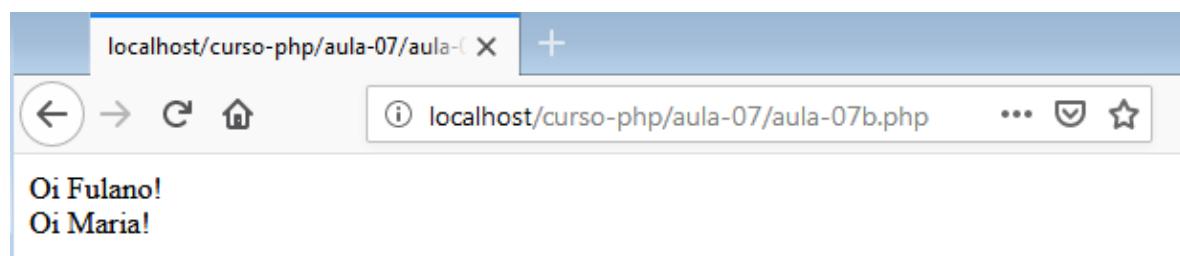
## Classe anônima (novas instâncias)

### aula-07/aula-07b.php

```
<?php

$pessoa = new class{
    public $nome = 'Fulano';
    public function oi(){
        return "Oi {$this->nome}!";
    }
};

echo $pessoa->oi() . "<br>";
$p1 = new $pessoa();
$p1->nome = 'Maria';
echo $p1->oi() . "<br>";
```



## Classe anônima (via function)

### aula-07/aula-07c.php

```
<?php

function pessoa_oo() {
    return new class {
        public $nome = 'Maria';
        public $cpf = '754';
        public function getNome(){
            return strtoupper($this->nome);
        }
    };
}

$pessoa = pessoa_oo();
echo $pessoa->nome . "<br>";
echo $pessoa->cpf. "<br>";
echo $pessoa->getNome();
```

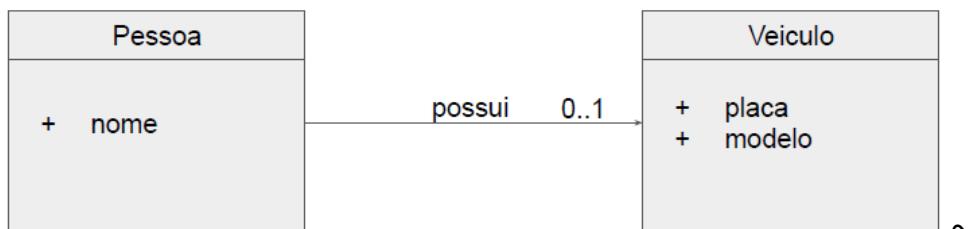
```

Maria
754
MARIA

```

## Associação entre objetos

Na programação orientada a objetos, a associação define um relacionamento entre classes de objetos que permite que uma instância de objeto faça com que outra execute uma ação em seu nome. Essa relação é estrutural, porque especifica que objetos de um tipo estão conectados a objetos de outro e não representam comportamento.



## aula-07/aula-07d.php

```

<?php

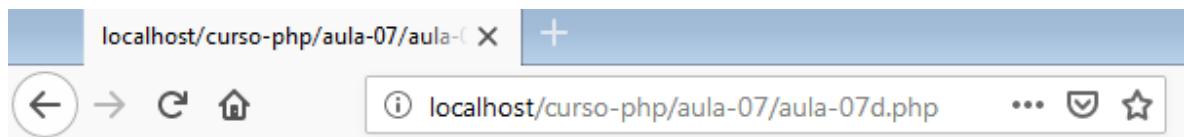
class Pessoa {
    public $nome;
    public $veiculo;
}

class Veiculo{
    public $placa;
    public $modelo;
}

$p1 = new Pessoa();
$p1->nome = 'Maria';
$p1->veiculo = new Veiculo(); // Aqui é criado o vínculo entre pessoa e veículo
$p1->veiculo->placa = 'ABC123';
$p1->veiculo->modelo = 'Tesla X1';

echo "<pre>";
print_r($p1);
echo "</pre>";

```



```
Pessoa Object
(
    [nome] => Maria
    [veiculo] => Veiculo Object
        (
            [placa] => ABC123
            [modelo] => Tesla X1
        )
)
```

### Associação entre objetos

```
1 <?php
2
3 class Pessoa {
4     public $nome;
5     public $veiculo;
6 }
7
8 class Veiculo{
9     public $placa;
10    public $modelo;
11 }
12
13 $p1 = new Pessoa();
14 $p1->nome = 'Maria';
15 $p1->veiculo = new Veiculo();
16
17 $p1->veiculo->placa = 'ABC123';
18 $p1->veiculo->modelo = 'Tesla X1';
19
20 var_dump($p1);
21
```

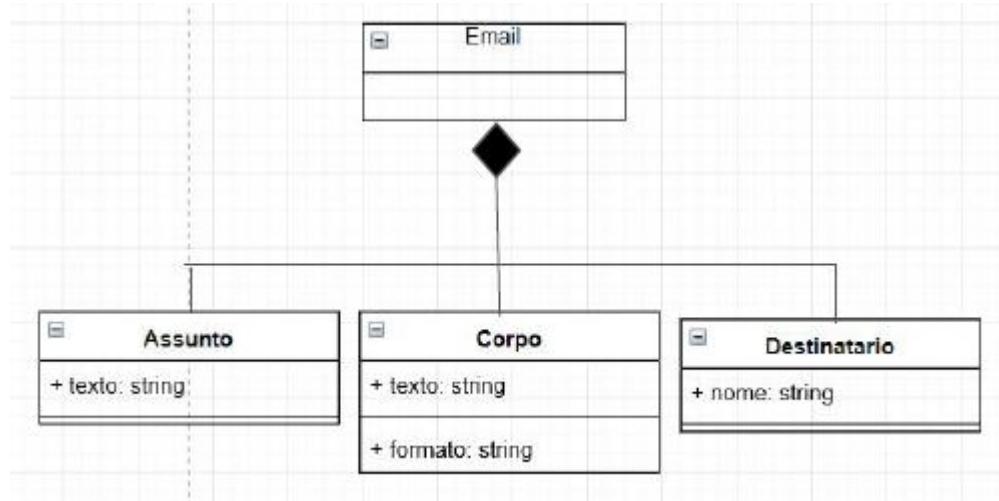
eval(); or quick preview in 8.0.3

Output for 8.0.3 | released 2021-03-04 | took 18 ms, 16.78 MiB

```
object(Pessoa)#1 (2) {
    ["nome"]=>
    string(5) "Maria"
    ["veiculo"]=>
    object(Veiculo)#2 (2) {
        ["placa"]=>
        string(6) "ABC123"
        ["modelo"]=>
        string(8) "Tesla X1"
    }
}
```

## Composição

- Composição é um tipo de associação onde o objeto composto é o único responsável pela disposição das partes componentes.
- O relacionamento entre o composto e o componente é um relacionamento forte "tem um", pois o objeto composto apropria-se do componente.
- Isso significa que o composto é responsável pela criação e destruição das partes componentes. Um objeto só pode fazer parte de um composto. Se o objeto composto for destruído, todas as partes componentes devem ser destruídas.
- A composição impõe o **encapsulamento**, pois as partes do componente geralmente são membros do objeto composto.
- A composição é caracterizada na UML pelo uso do losango preenchido.



## **aula-07/aula-07e.php**

```
<?php

class Assunto {
    public $texto;
}

class Corpo {
    public $texto;
    public $formato;
}

class Destinatario {
    public $nome;
}

class Email {
    private $assunto;
    private $corpo;
    private $destinatario;

    public function __construct() {
        $this->assunto = new Assunto();
        $this->corpo = new Corpo();
        $this->destinatario = new Destinatario();
    }
}

$email1 = new Email();
echo "<pre>";
print_r($email1);
echo "</pre>";
```

```
Email Object
(
    [assunto:Email:private] => Assunto Object
        (
            [texto] =>
        )

    [corpo:Email:private] => Corpo Object
        (
            [texto] =>
            [formato] =>
        )

    [destinatario:Email:private] => Destinatario Object
        (
            [nome] =>
        )
)
```

## Composição

```
1 <?php
2
3 class Assunto {
4     public $texto;
5 }
6
7 class Corpo {
8     public $texto;
9     public $formato;
10}
11
12 class Destinatario {
13     public $nome;
14 }
15
16 class Email {
17     private $assunto;
18     private $corpo;
19     private $destinatario;
20
21     public function __construct() {
22         $this->assunto = new Assunto();
23         $this->corpo = new Corpo();
24         $this->destinatario = new Destinatario();
25     }
26
27     public function setAssunto($texto) {
28         $this->assunto->texto = $texto;
29     }
30
31     public function getAssunto() {
32         return $this->assunto->texto();
33     }
34 }
35
36
37 $email1 = new Email();
38 print_r($email1);
39
```

Output for 8.0.3 | released 2021-03-04 | took 23 ms, 16.77 MiB

```
Email Object
(
    [assunto:Email:private] => Assunto Object
        (
            [texto] =>
        )

    [corpo:Email:private] => Corpo Object
        (
            [texto] =>
            [formato] =>
        )

    [destinatario:Email:private] => Destinatario Object
        (
            [nome] =>
        )
)
```

## Composição (versão com classes anônimas)

### aula-07/aula-07f.php

```
<?php

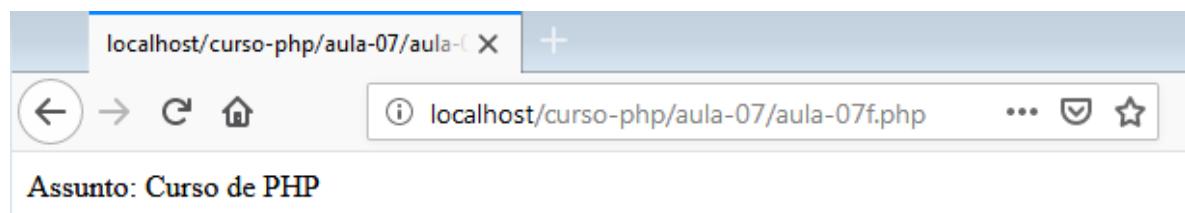
class Email {
    private $assunto;
    private $corpo;
    private $destinatario;

    public function __construct() {
        $this->assunto = new class {
            public $texto;
        };
        $this->corpo = new class {
            public $texto;
            public $formato;
        };
        $this->destinatario = new class {
            public $nome;
        };
    }

    public function setAssunto($texto){
        $this->assunto->texto = $texto;
    }

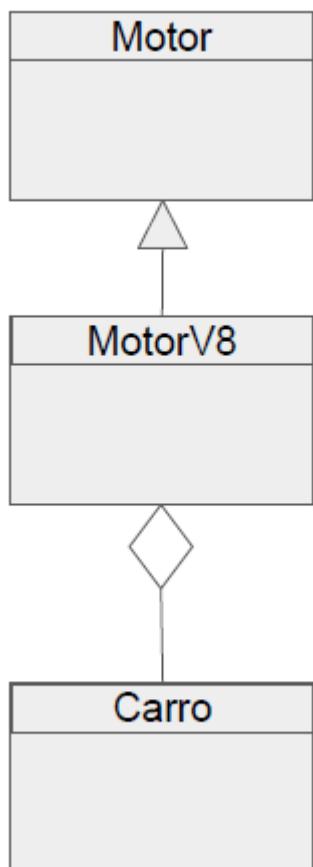
    public function getAssunto(){
        return $this->assunto->texto;
    }
    //implementar os próximos getters & setters
}

$email = new Email();
$email->setAssunto('Curso de PHP');
echo "Assunto: " . $email->getAssunto();
```



## Agregação

- Agregação é um tipo de associação que especifica um relacionamento de todo / parte entre a parte agregada (todo) e o componente.
- Esse relacionamento entre o agregado e o componente é um relacionamento "tem-um" fraco, pois o componente pode sobreviver ao objeto agregado.
- O objeto componente pode ser acessado através de outros objetos sem passar pelo objeto agregado. O objeto agregado não participa do ciclo de vida do objeto de componente, o que significa que o objeto de componente pode sobreviver ao objeto agregado. O estado do objeto componente ainda faz parte do objeto agregado.
- A agregação é caracterizada na UML pelo uso do losango sem vazio.



## **aula-07/aula-07g.php**

```
<?php

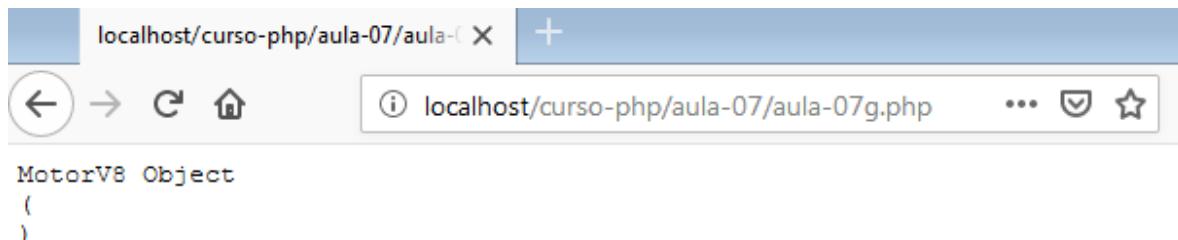
abstract class Motor {}

class MotorV8 extends Motor {}

class Carro {
    private $motor;
    public function __construct(Motor $motor) {
        $this->motor = $motor;
    }
}

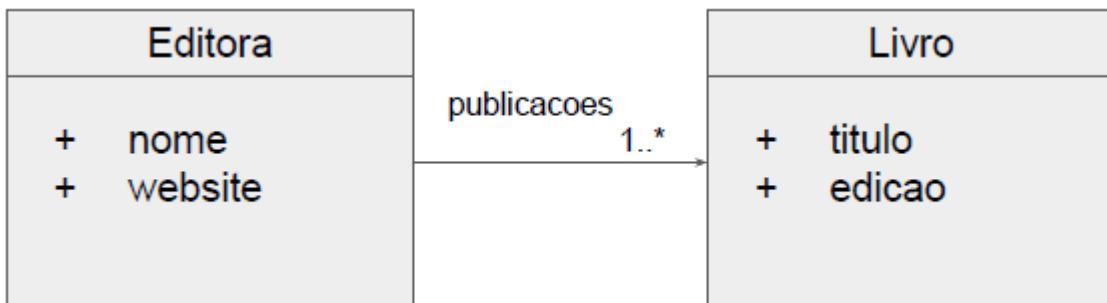
$motorV8 = new MotorV8();
$carro1 = new Carro($motorV8);
unset($carro1);

echo "<pre>";
print_r($motorV8);
echo "</pre>";
```



## Associação (1..\*)

Uma associação um-para-muitos (ou têm muitos) é um tipo de associação que estabelece uma vinculação onde uma instância da classe do lado esquerdo tem zero ou mais instâncias de outra classe. Para representar esta multiplicidade, podemos utilizar uma coleção (ex.: array) para “armazenar” / “intermediar” esta relação.



### aula-07/aula-07h.php

```
<?php

class Livro {
    public $titulo;
    public $edicao;
}

class Editora {
    public $nome;
    public $website;
    public $publicacoes = [];
}

$livro1 = new Livro();
$livro1->titulo = 'Aventuras do PHP';
$livro2 = new Livro();
$livro2->titulo = 'A vida';

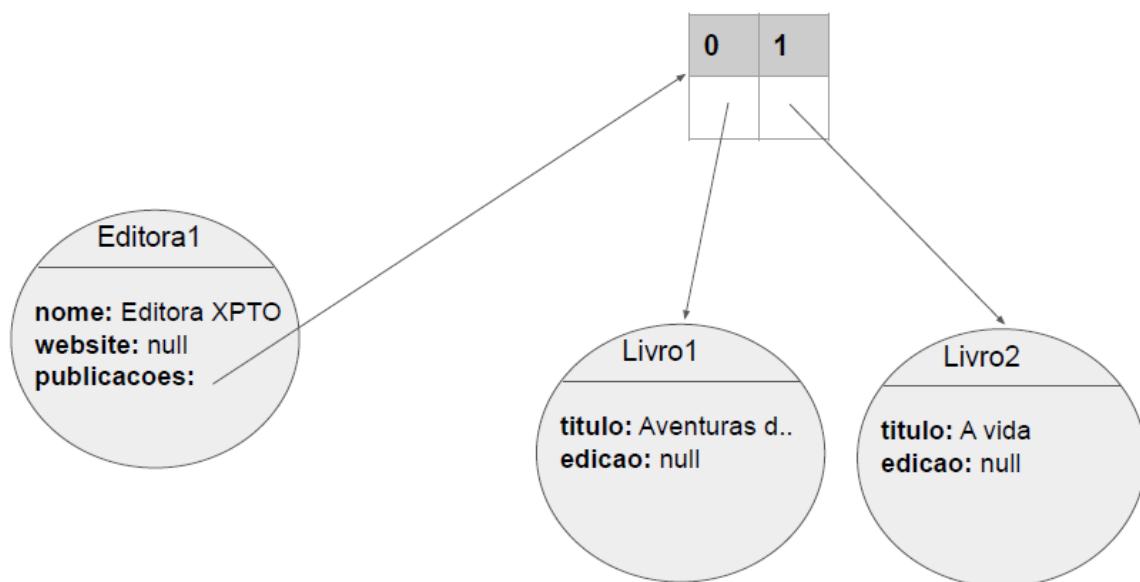
$editora1 = new Editora();
$editora1->nome = 'Editora XPTO';
$editora1->publicacoes[] = $livro1;
$editora1->publicacoes[] = $livro2;

foreach($editora1->publicacoes as $livro){
    echo $livro->titulo . "<br>";
}
```

Aventuras do PHP  
A vida

localhost/curso-php/aula-07/aula-07h.php

## Associação (1..\*)



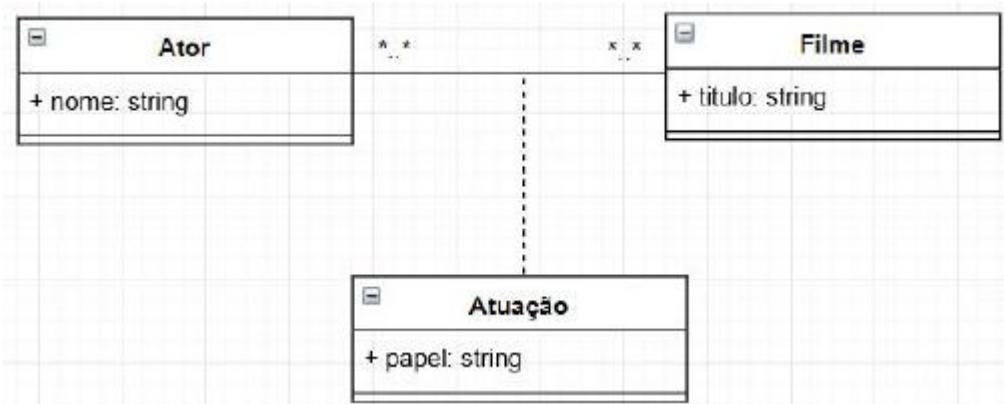
## Classe Associativa

Uma classe associativa é uma classe que faz parte de um relacionamento de associação entre duas outras classes.

É possível “anexar” uma classe associativa a um relacionamento para fornecer informações adicionais sobre o relacionamento. Uma classe de associação é idêntica a outras classes e pode conter operações, atributos e outras associações.

Por exemplo, um Ator pode atuar em muitos Filmes e um Filme pode contar com a atuação de vários Atores. Em cada Filme, o Ator pode atuar com um papel distinto. Por isso, neste caso, uma classe associativa chamada Atuação pode definir melhor esta associação fornecendo informação adicional do papel que o Ator desempenhou em um determinado Filme.

Como a figura a seguir ilustra, uma classe de associação é conectada a uma associação por uma linha pontilhada.



## aula-07/aula-07i.php

```
<?php

class Ator {
    public $nome;
    public $atuacoes = [];
}

class Atuacao {
    public $ator;
    public $filme;
    public $papel;

    public function __construct(Ator $ator, Filme $filme, string $papel){
        $this->ator = $ator;
        $this->filme = $filme;
        $this->papel = $papel;
    }
}

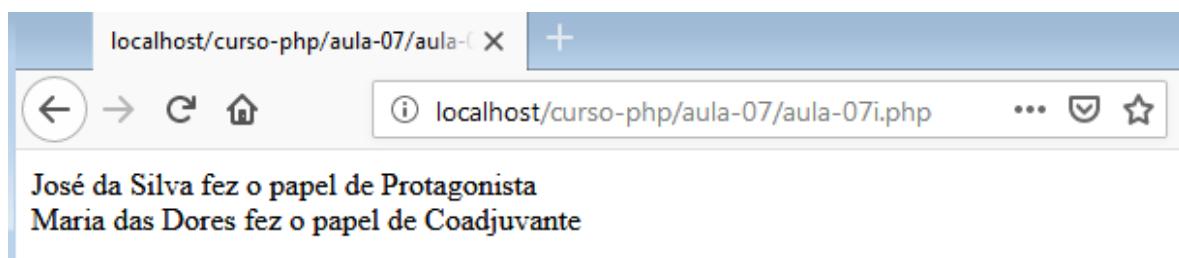
class Filme {
    public $título;
    public $atuacoes = [];
}

$ator1 = new Ator();
$ator1->nome = 'José da Silva';

$ator2 = new Ator();
$ator2->nome = 'Maria das Dores';

$filme1 = new Filme();
$filme1->título = 'Aventuras do barulho';
```

```
$filme1->atuacoes[] =  
new Atuacao($ator1, $filme1, 'Protagonista');  
  
$filme1->atuacoes[] =  
new Atuacao($ator2, $filme1, 'Coadjuvante');  
  
foreach($filme1->atuacoes as $atuacao){  
    echo $atuacao->ator->nome . " fez o papel de " . $atuacao->papel . "<br>";  
}
```



# Aula 08 - Reflection, Namespace e Autoload

## Autoload (carregamento automático de classes)

### Autoload (Carregamento automático de Classes)

- Quando trabalhamos com mais de uma Classe, Interface e/ou Trait, é interessante utilizar algum recurso que possibilite o carregamento dinâmico/automático das estruturas em seus respectivos arquivos \*.php.
- O PHP possui duas funções de registro automático: o `__autoload()` e o `spl_autoload_register()`
- O `__autoload()` foi depreciado à partir do PHP 7.2.
- O `spl_autoload_register()` permite que vários carregadores automáticos sejam registrados, que serão executados por sua vez até que uma classe, interface ou trait correspondente seja localizada e carregada, ou até que todas as opções de carregamento automático tenham sido esgotadas. Isso significa que, se você estiver usando algum framework ou outras bibliotecas que implementam seus próprios carregadores automáticos, não é necessário se preocupar com a possibilidade de conflitos.

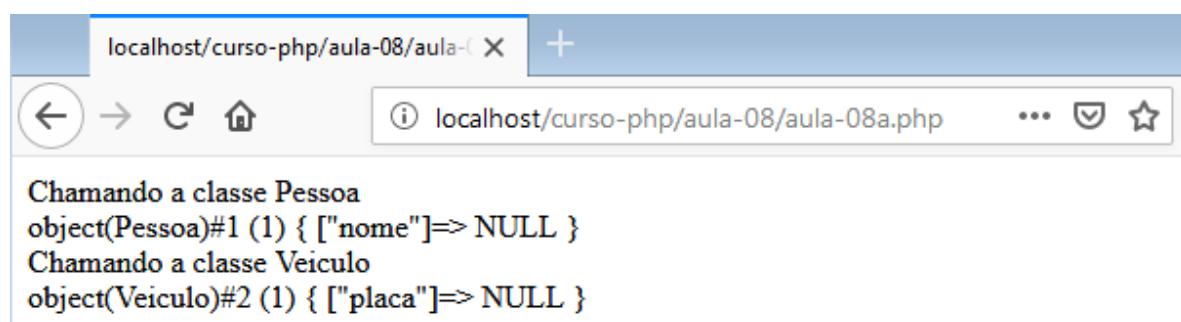
### aula-08/aula-08a.php

```
<?php

function meu_autoloader($class) {
    echo "Chamando a classe " . $class . "<br>";
    include_once("classes/" . $class . ".php");
}

spl_autoload_register('meu_autoloader');

$pessoa1 = new Pessoa();
var_dump($pessoa1);
echo "<br>";
$veiculo1 = new Veiculo();
var_dump($veiculo1);
```



## Reflection

### Reflection

- Permite introspecção no código Orientado a objeto.
- O Reflection auxilia os desenvolvedores a criar bibliotecas genéricas de software para exibir dados, processar diferentes formatos de dados, realizar serialização, agrupar dados e muito mais.
- O Reflection pode ser usado para observar e modificar a execução do programa em tempo de execução.
- Em linguagens de programação orientada a objeto, como PHP, o Reflection permite a inspeção de classes, interfaces, campos e métodos em tempo de execução sem conhecer os nomes das interfaces, campos, métodos em tempo de execução.
- Também permite a instanciação de novos objetos e a invocação de métodos de forma dinâmica.

### aula-08/aula-08b.php

```
<?php
class Animal {}

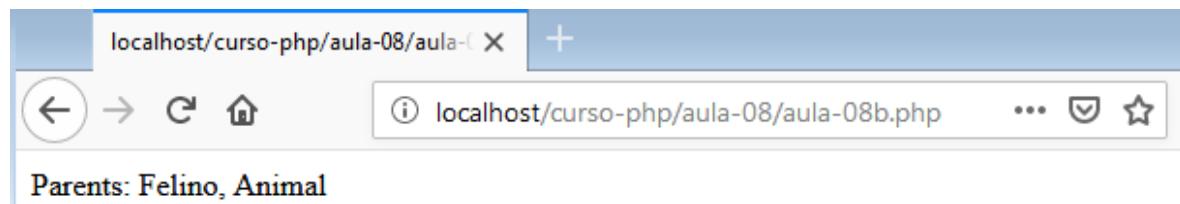
class Felino extends Animal {}

class Gato extends Felino {}

$class = new ReflectionClass(Gato::class);

while ($parent = $class->getParentClass()) {
    $parents[] = $parent->getName();
    $class = $parent;
}

echo "Parents: " . implode(", ", $parents);
```



## Reflection (class)

### aula-08/aula-08c.php

```
<?php

class Aluno {
    public $nome;
    protected $endereco;
    private $telefone;
}

$aluno1 = new Aluno();
$reflect = new ReflectionClass($aluno1);
$props = $reflect->getProperties(ReflectionProperty::IS_PUBLIC | 
ReflectionProperty::IS_PROTECTED);

foreach ($props as $prop) {
    print $prop->getName() . "<br>";
}

echo "<pre>";
var_dump($props);
echo "</pre>";
```

```
localhost/curso-php/aula-08/aula-08c.php +  
← → ⌂ ⌂ ⌂ ⓘ localhost/curso-php/aula-08/aula-08c.php ... 🌐 ☆  
  
nome  
endereco  
  
array(2) {  
    [0]=>  
    object(ReflectionProperty)#3 (2) {  
        ["name"]=>  
        string(4) "nome"  
        ["class"]=>  
        string(5) "Aluno"  
    }  
    [1]=>  
    object(ReflectionProperty)#4 (2) {  
        ["name"]=>  
        string(8) "endereco"  
        ["class"]=>  
        string(5) "Aluno"  
    }  
}
```

## Reflection (alterando visibilidade)

Com o uso do Reflection é possível “modificar” a visibilidade de um atributo ou método em tempo de execução através do setAccessible(true); Entretanto, o acesso aos valores/retornos precisam ser feitos através dos métodos getValue() ou invoke().

No caso dos atributos é possível utilizar o setValue() para modificar o valor de um atributo também.

### aula-08/aula-08d.php

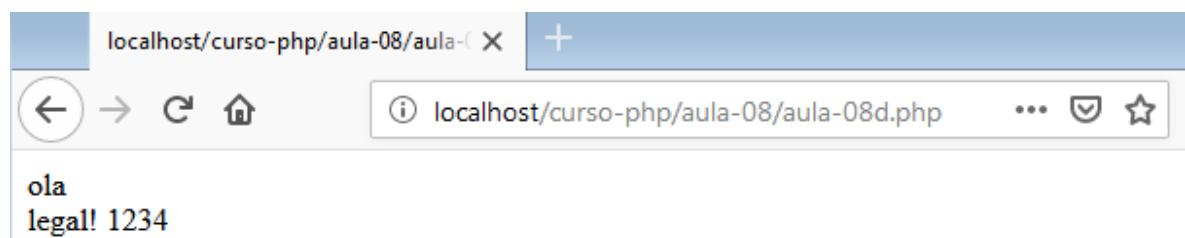
```
<?php

class Teste {
    private $propriedade = 'ola';
    private function dizSegredo(){
        echo $this->propriedade.'! 1234';
    }
}

$class = new ReflectionClass("Teste");
$property = $class->getProperty("propriedade");
$property->setAccessible(true);

$teste1 = new Teste();
//echo $teste1->propriedade; // Não funciona aqui
echo $property->getValue($teste1) . "<br>";
$property->setValue($teste1, 'legal');

$method = new ReflectionMethod("Teste", 'dizSegredo');
$method->setAccessible(true);
echo $method->invoke($teste1) . "<br>";
```



## Reflection (gerar instâncias e invocar métodos)

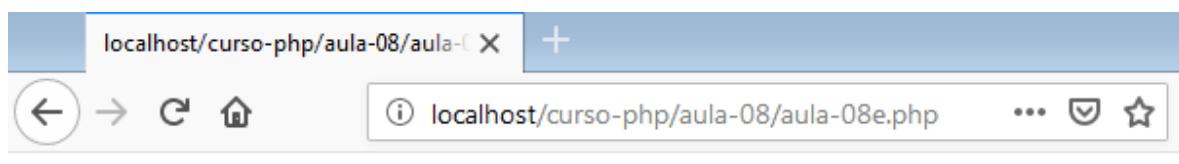
### aula-08/aula-08e.php

```
<?php

class Pessoa {
    public function digaOi(){
        return 'oi!';
    }
}

//com reflection
$reflector = new ReflectionClass('Pessoa');
$instance = $reflector->newInstance();
var_dump($instance);

echo "<br>";
$metodo = $reflector->getMethod('digaOi');
echo $metodo->invoke($instance);
```



## Reflection (recuperar comentários)

Com Reflection também é possível recuperar comentários das classes e métodos através do getDocComment().

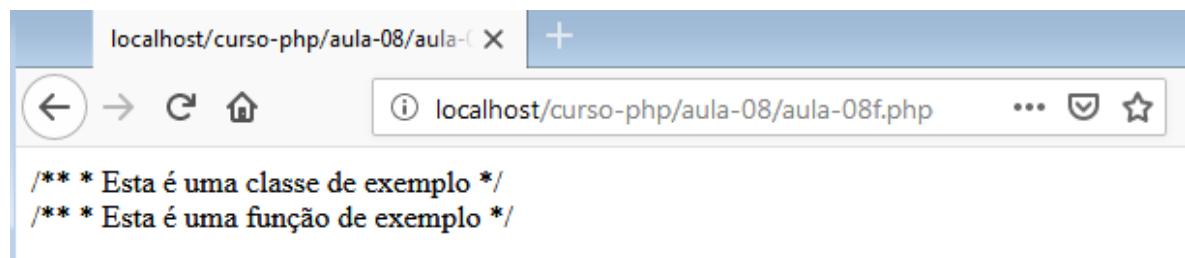
### aula-08/aula-08f.php

```
<?php

/**
 * Esta é uma classe de exemplo
 */

class Exemplo {
    /**
     * Esta é uma função de exemplo
     */
    public function fazNada(){}
}

$reflector = new ReflectionClass(Exemplo::class);
echo $reflector->getDocComment() . "<br>";
echo $reflector->getMethod('fazNada')->getDocComment();
```



## Namespace

### Namespace

- Os namespaces são basicamente uma maneira de organizar suas classes PHP e evitar conflitos de código.
- Usamos a palavra reservada **namespace**
- Namespace não estão relacionados diretamente com a estrutura de diretórios.
- Namespace usam “backslash” (\) como separador
- É possível referenciar a classe usando o **use** para evitar escrever o namespace completo novamente
- Apelidos e Conflitos podem ser resolvidos com o uso do **as**

### Namespace

Pessoa.php

exemplo.php

<?php

<?php

**namespace** Modelo;

**require** "Pessoa.php";

**class Pessoa {}**

// não funciona !

//\$pessoa1 = new Pessoa();

\$pessoa1 = **new** Modelo\Pessoa();

## Namespace (modelos homônimos)

### aula-08/modelos/fruta/Manga.php

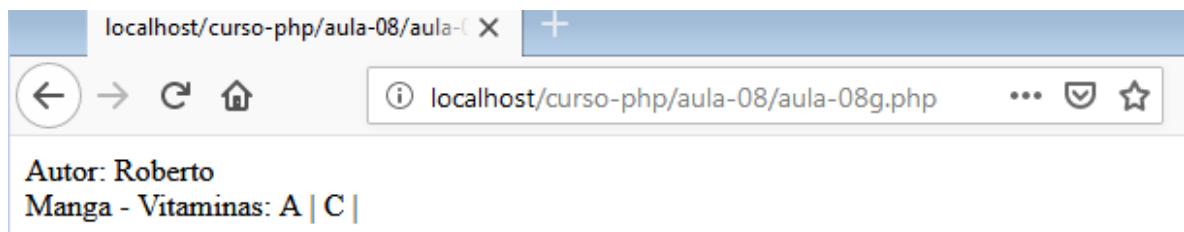
```
<?php  
  
namespace Modelo\Fruta;  
  
class Manga {  
    public $vitaminas = ['A','C'];  
}  
  
?>
```

### aula-08/modelos/literatura/Manga.php

```
<?php  
  
namespace Modelo\Literatura;  
  
class Manga {  
    public $autor;  
    public $editora;  
}  
  
?>
```

### aula-08/aula-08g.php

```
<?php  
  
require "Modelo\Fruta\Manga.php";  
require "Modelo\Literatura\Manga.php";  
  
$manga1 = new Modelo\Literatura\Manga();  
$manga1->autor = "Roberto";  
echo "Autor: " . $manga1->autor . "<br>";  
  
$manga2 = new Modelo\Fruta\Manga();  
  
echo "Manga - Vitaminas: ";  
foreach($manga2->vitaminas as $vitamina){  
    echo $vitamina . " | ";  
}
```

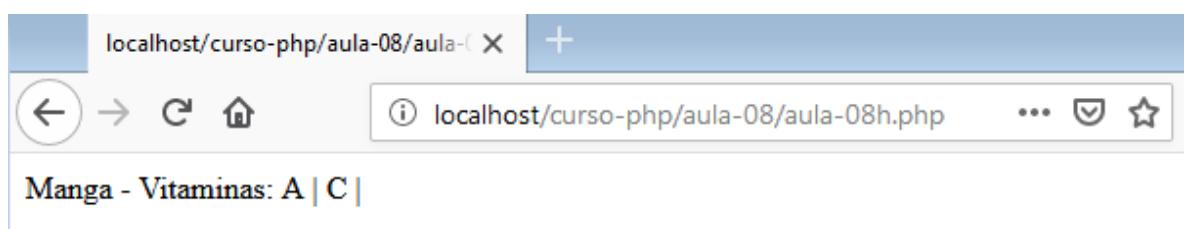


## Namespace (use)

Com a palavra reservada use, é possível invocar o nome completo de uma classe para que a mesma possa ser utilizada sem a necessidade do namespace após o use.

### aula-08/aula-08h.php

```
<?php  
  
require "Modelo\Fruta\Manga.php";  
  
use Modelo\Fruta\Manga;  
  
$manga2 = new Manga();  
  
echo "Manga - Vitaminas: ";  
foreach($manga2->vitaminas as $vitamina){  
    echo $vitamina . " | ";  
}  
}
```



## Namespace (resolução de conflitos)

### aula-08/aula-08i.php

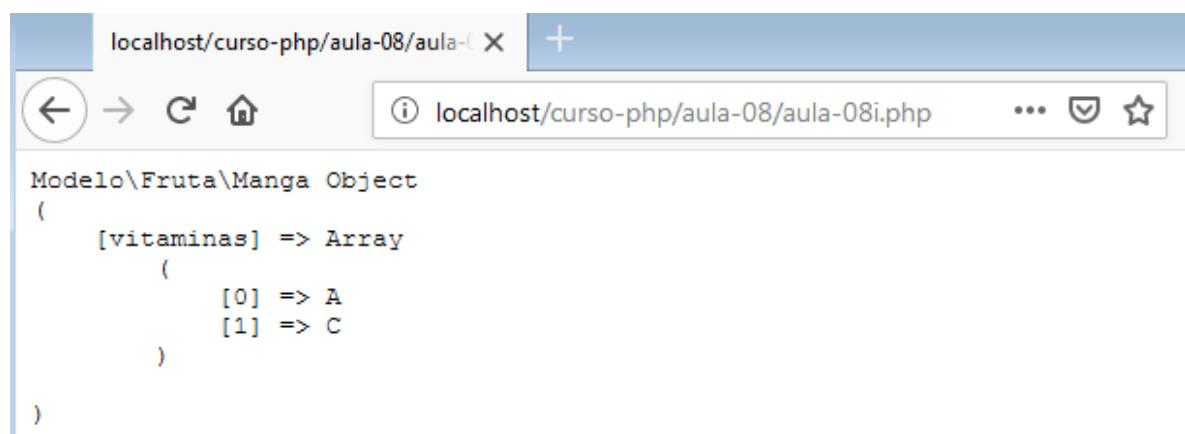
```
<?php

require "Modelo\Fruta\Manga.php";
require "Modelo\Literatura\Manga.php";

use Modelo\Fruta\Manga as FrutaManga;
use Modelo\Literatura\Manga;

$manga1 = new Manga();
$manga2 = new FrutaManga();

echo "<pre>";
print_r($manga2);
echo "</pre>";
```



```
localhost/curso-php/aula-08/aula-08i.php +  
← → ⌂ ⌂ ... 🌐 ☆  
① localhost/curso-php/aula-08/aula-08i.php  
  
Modelo\Fruta\Manga Object  
(  
    [vitaminas] => Array  
        (  
            [0] => A  
            [1] => C  
        )  
)  
)
```

## Namespace + Autoloader

O nome do namespace (e consonância com o nome dos diretórios) pode ser utilizado para incluir automaticamente as classes no código. Isto não exime o desenvolvedor de utilizar o use ou nome completo da classe (FQN - Fully Qualified Name).

### aula-08/aula-08j.php

```
<?php

function meu_autoloader($class) {
    include_once $class . '.php';
}

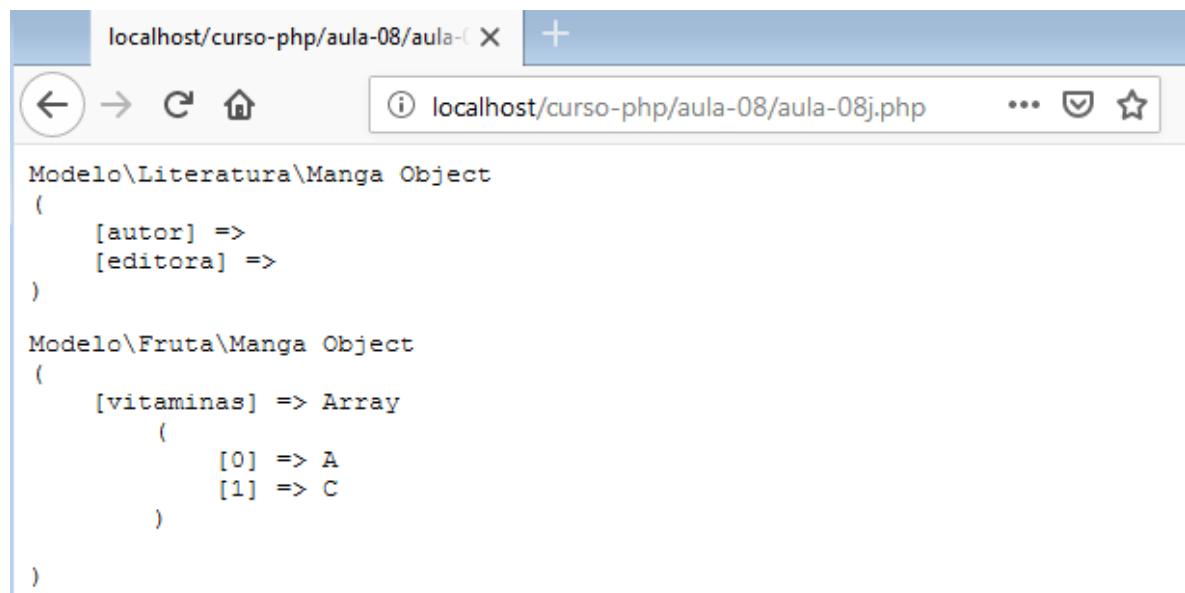
spl_autoload_register('meu_autoloader');

use modelo\fruta\Manga as FrutaManga;
use modelo\literatura\Manga;

$manga1 = new Manga();
$manga2 = new FrutaManga();

echo "<pre>";
print_r($manga1);
echo "</pre>";

echo "<pre>";
print_r($manga2);
echo "</pre>";
```



```
localhost/curso-php/aula-08/aula-08j.php +  
← → ⌂ ⌂ ⓘ localhost/curso-php/aula-08/aula-08j.php ... ⌂ ☆  
  
Modelo\Literatura\Manga Object  
(  
    [autor] =>  
    [editora] =>  
)  
  
Modelo\Fruta\Manga Object  
(  
    [vitaminas] => Array  
        (  
            [0] => A  
            [1] => C  
        )  
)
```

## Namespace (Group use PHP >= 7.0)

### aula-08/aula-08k.php

```
<?php

function meu_autoloader($class) {
    include_once $class . '.php';
}

spl_autoload_register('meu_autoloader');

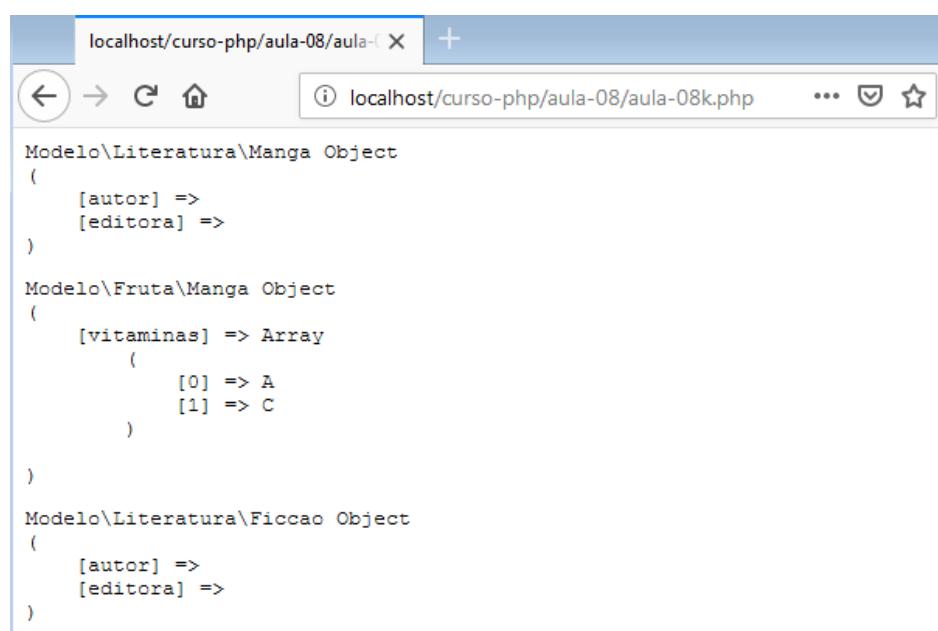
use modelo\fruta\Manga as FrutaManga;
use modelo\literatura\{Conto, Manga, Ficcao};

$manga1 = new Manga();
$manga2 = new FrutaManga();
$ficcao1 = new Ficcao();

echo "<pre>";
print_r($manga1);
echo "</pre>";

echo "<pre>";
print_r($manga2);
echo "</pre>";

echo "<pre>";
print_r($ficcao1);
echo "</pre>";
```



```
localhost/curso-php/aula-08/aula-08k.php

Modelo\Literatura\Manga Object
(
    [autor] =>
    [editora] =>
)

Modelo\Fruta\Manga Object
(
    [vitaminas] => Array
        (
            [0] => A
            [1] => C
        )
)

Modelo\Literatura\Ficcao Object
(
    [autor] =>
    [editora] =>
)
```

# Aula 09 - Iterator yield, Typed Properties, DateTime, StdClass

## aula-09/aula-09a.php

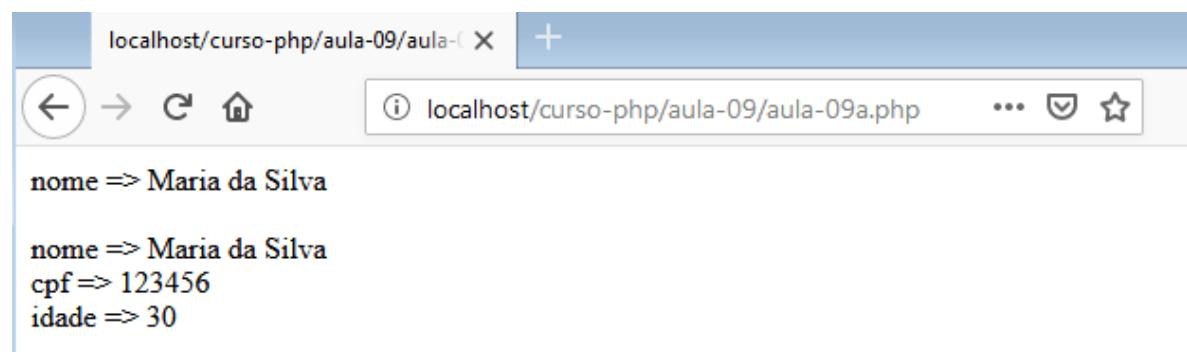
```
<?php

class Pessoa {
    public $nome = 'Maria da Silva';
    private $cpf = '123456';
    protected $idade = 30;

    function iterar() {
        foreach ($this as $key => $value) {
            print "$key => $value<br>";
        }
    }
}

$pessoa = new Pessoa();

//Aqui só é possível iterar os atributos públicos
foreach($pessoa as $key => $value) {
    print "$key => $value<br>";
}
echo "<br>";
//no método usando o $this é possível iterar todos os valores
$pessoa->iterar();
```



## Interface Iterator

### Interface Iterator

- Interface Iterator possui um conjunto abstrato de métodos para iteradores externos ou objetos que podem ser iterados internamente.
- Quando uma nova classe implementa esta interface, suas instâncias tornam-se iteráveis, similar ao uso de um array.

|   |
|---|
| <<interface>>   |
| Iterator  |
| + current()<br>+ key()<br>+ next()<br>+ rewind()<br>+ valid() |

Quando um objeto implementa o Iterator, o mesmo torna-se passível de ser “iterado” de forma transparente, similar a um array, por uma estrutura de laço dinâmica como o “foreach”, graças aos métodos que “garantem” o processo da iteração.

```
$sequencia = new Fibonacci();
foreach($sequencia as $value){
    echo "valor: $value" . "<br>";
}
```

## **aula-09/aula-09b.php**

```
<?php

class Fibonacci implements Iterator {
    private $anterior = 1;
    private $atual = 0;
    private $key = 0;

    public function __construct (int $max = 100) {
        $this->max = $max;
    }

    public function current() {
        return $this->atual;
    }

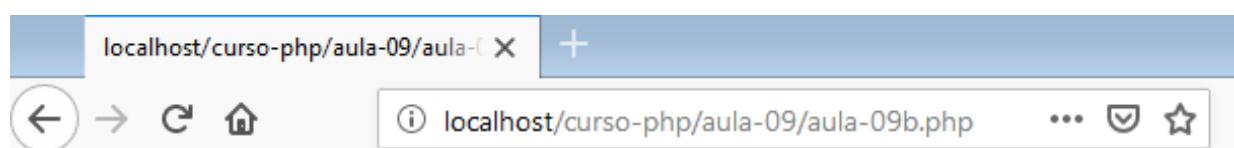
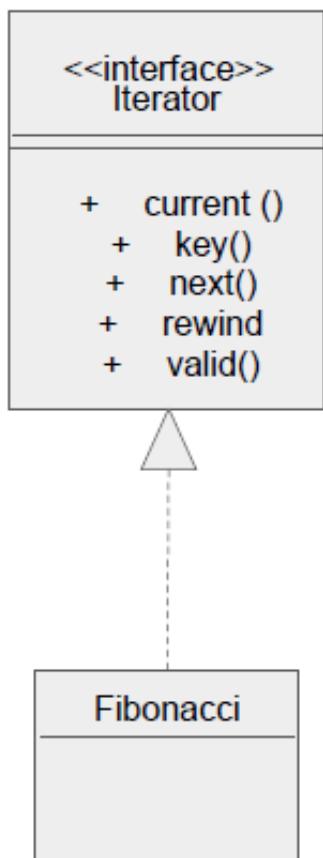
    public function key() {
        return $this->key;
    }

    public function next() {
        $novo_atual = $this->atual;
        $this->atual += $this->anterior;
        $this->anterior = $novo_atual;
        $this->key++;
    }

    public function rewind() {
        $this->anterior = 1;
        $this->atual = 0;
        $this->key = 0;
    }

    public function valid() {
        if($this->atual > $this->max)
            return false;
        return true;
    }
}

$sequencia = new Fibonacci();
foreach($sequencia as $value){
    echo "valor: $value" . "<br>";
}
```



```
valor: 0
valor: 1
valor: 1
valor: 2
valor: 3
valor: 5
valor: 8
valor: 13
valor: 21
valor: 34
valor: 55
valor: 89
```

## Generator

Quando uma função contém a palavra reservada `yield`, o PHP automaticamente interpreta a mesma como um objeto do tipo Generator. Similar ao uso do Iterator, um generator permite que o objeto possa ser iterado. Além disso, o `yield` “pausa” o fluxo natural do código da função e permite retorna um valor similar ao `return`.

### aula-09/aula-09c.php

```
<?php

function meuGenerator() {
    echo "Um".<br>;
    yield 1;
    echo "Dois".<br>;
    yield 2;
    echo "Três".<br>;
    yield 3;
}

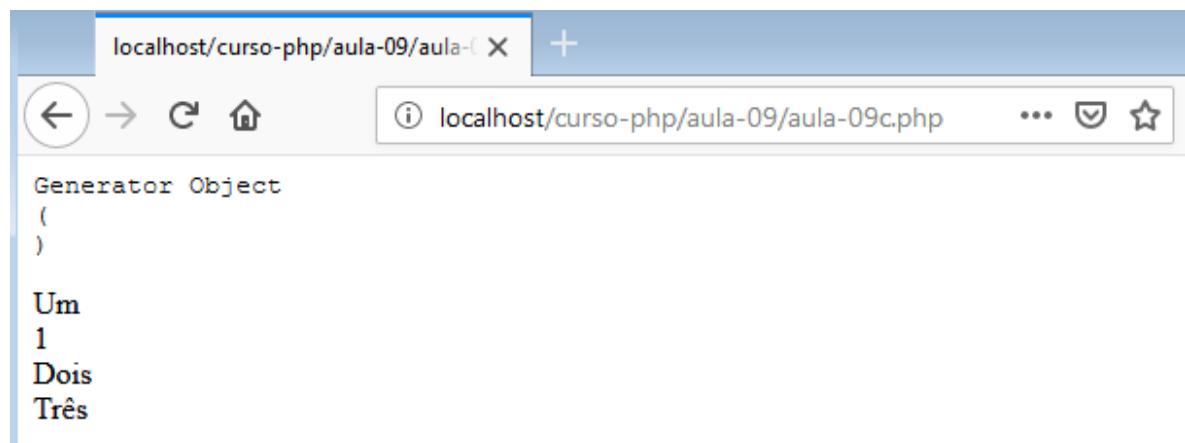
$value = meuGenerator();

echo "<pre>";
print_r($iterator);
echo "</pre>";

$value = $iterator->current();

echo $value . "<br>";

$value = $iterator->next();
$value = $iterator->next();
```



## Generator (yield from)

Com o uso yield from é possível recuperar os valores de outros generators, iteradores ou arrays.

### aula-09/aula-09d.php

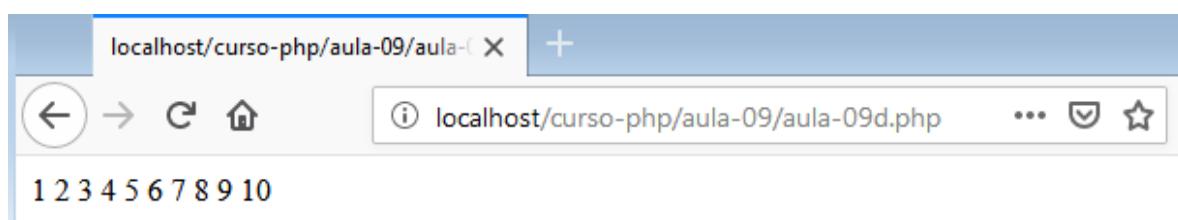
```
<?php

function contar10() {
    yield 1;
    yield 2;
    yield from [3, 4];
    yield from new ArrayIterator([5, 6]);
    yield from sete_oito();
    yield 9;
    yield 10;
}

function sete_oito() {
    yield 7;
    yield from oito();
}

function oito() {
    yield 8;
}

foreach (contar10() as $num) {
    echo "$num ";
}
```



## Iterator to Array

Com o uso da função iterator\_to\_array é possível converter um objeto iterável para array

### aula-09/aula-09e.php

```
<?php

class Fibonacci implements Iterator {
    private $anterior = 1;
    private $atual = 0;
    private $key = 0;

    public function __construct (int $max = 100) {
        $this->max = $max;
    }

    public function current() {
        return $this->atual;
    }

    public function key() {
        return $this->key;
    }

    public function next() {
        $novo_atual = $this->atual;
        $this->atual += $this->anterior;
        $this->anterior = $novo_atual;
        $this->key++;
    }

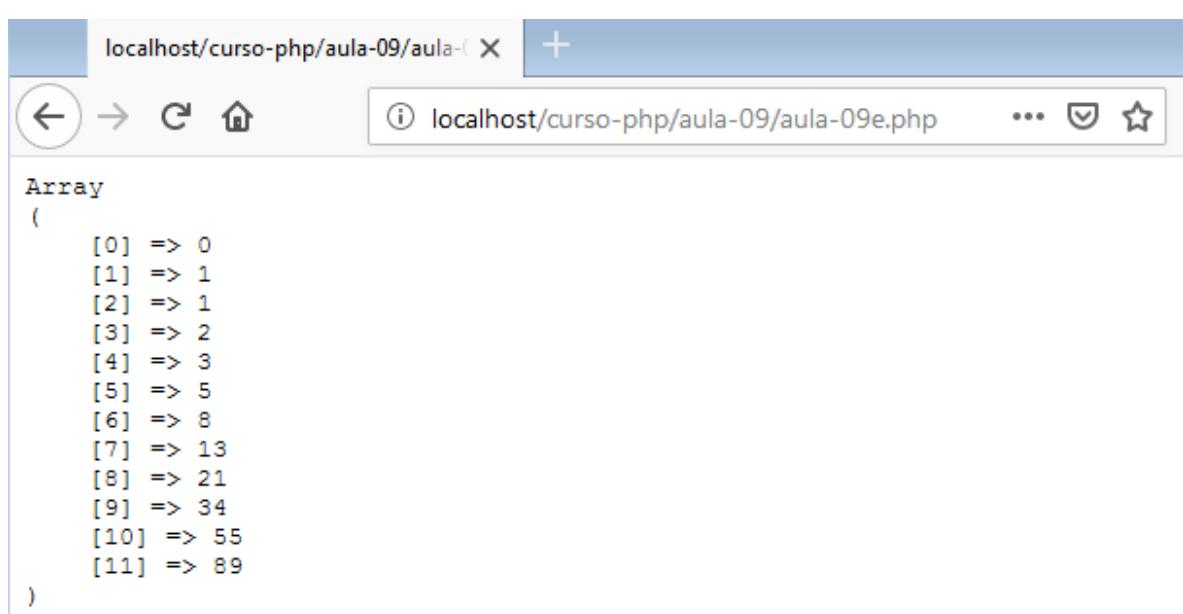
    public function rewind() {
        $this->anterior = 1;
        $this->atual = 0;
        $this->key = 0;
    }

    public function valid() {
        if($this->atual > $this->max)
            return false;
        return true;
    }
}

$seq = new Fibonacci();
```

```
$array_fibonnaci = iterator_to_array($seq);

echo "<pre>";
print_r($array_fibonnaci);
echo "</pre>";
```



A screenshot of a web browser window. The address bar shows the URL `localhost/curso-php/aula-09/aula-09e.php`. The page content displays an array of Fibonacci numbers from index 0 to 11.

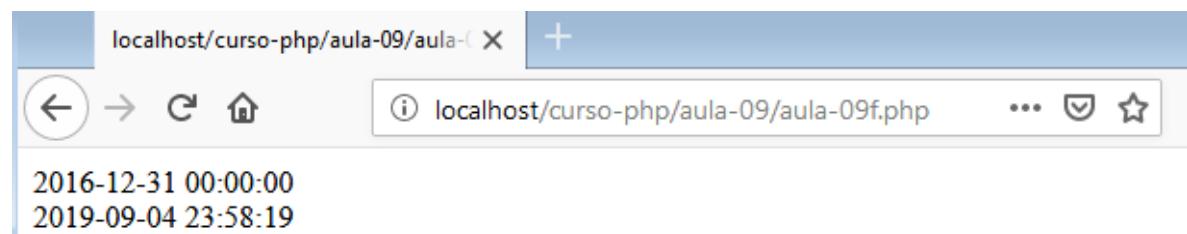
```
Array
(
    [0] => 0
    [1] => 1
    [2] => 1
    [3] => 2
    [4] => 3
    [5] => 5
    [6] => 8
    [7] => 13
    [8] => 21
    [9] => 34
    [10] => 55
    [11] => 89
)
```

## Classe DateTime

Classe DateTime é uma alternativa O.O. à função date(). Ela possui um conjunto de métodos que permitem realizar operações complexas com datas e horários. Além disso, por sua natureza de Classe, é possível estender a mesma e criar classes customizadas de Data e Hora.

### aula-09/aula-09f.php

```
<?php  
  
$dateTime = new DateTime('2016-12-31');  
echo $dateTime->format('Y-m-d H:i:s') . "<br>";  
  
$dateTimeAgora = new DateTime();  
echo $dateTimeAgora->format('Y-m-d H:i:s');
```

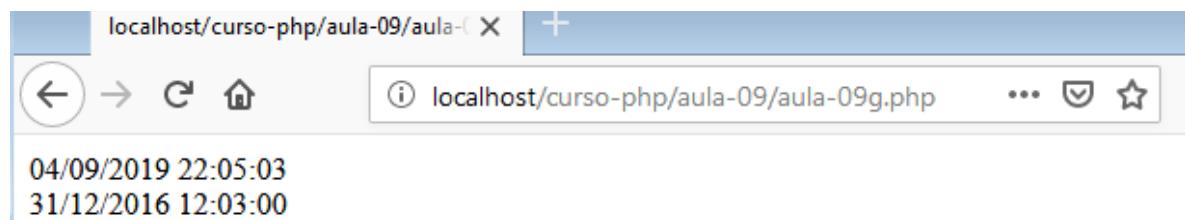


## DateTimeZone (Fuso horário)

É possível setar o fuso horário no PHP através da função `date_default_timezone_set()` passando como valor o fuso horário desejado. Também é possível determinar o fuso horário individualmente no momento da criação do objeto `DateTime` passando uma instância de `DateTimeZone`.

### aula-09/aula-09g.php

```
<?php  
  
date_default_timezone_set('America/Sao_Paulo');  
  
$dateTimeAgora = new DateTime();  
echo $dateTimeAgora->format('d/m/Y H:i:s') . "<br>";  
  
$dateTime = new DateTime('2016-12-31 12:03:00',  
new DateTimeZone('America/Sao_Paulo'));  
echo $dateTime->format('d/m/Y H:i:s') . "<br>";
```



## Classe DateTime (createFromFormat)

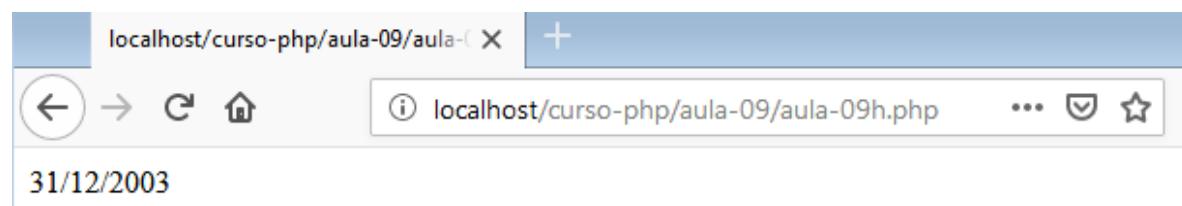
Classe DateTime (createFromFormat)

```
<?php  
  
$dateBr = DateTime::createFromFormat('d/m/Y', '31/12/2003');  
echo $dateBr->format('d/m/Y');
```

Formato para exibição (string)      Formato da data      Data no formato

### aula-09/aula-09h.php

```
<?php  
  
$dateBr = DateTime::createFromFormat('d/m/Y', '31/12/2003');  
echo $dateBr->format('d/m/Y');
```



## Classe DateTime (modificando data)

### aula-09/aula-09i.php

```
<?php

echo "<p>Data: 31/01/2018</p>";

// Adicionar 1 dia

$dateTime = new DateTime('2018-01-31');
$dateTime->modify('+1 day');

echo "Adicionar 1 dia<br>";
echo $dateTime->format('d/m/Y');

echo "<br><br><hr>";

// Adicionar 10 dias úteis

echo "<p>Data: 04/09/2018</p>";

echo "Adicionar 10 dias úteis<br>";
$dateTime = new DateTime();
$dateTime->modify('+10 weekdays');
echo $dateTime->format('d/m/Y');

echo "<br><br>";

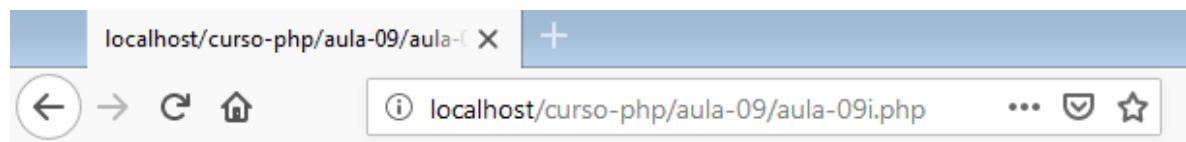
// Segunda-feira dessa semana

echo "Segunda-feira dessa semana<br>";
$dateTime = new DateTime();
$dateTime->modify('monday this week');
echo $dateTime->format('d/m/Y');

echo "<br><br>";

// Adicionar 1 mês - 3 dias

echo "Adicionar 1 mês - 3 dias<br>";
$dateTime = new DateTime();
$dateTime->modify('+1 month -3 days');
echo $dateTime->format('d/m/Y');
```



Data: 31/01/2018

Adicionar 1 dia

01/02/2018

---

Data: 04/09/2018

Adicionar 10 dias úteis

19/09/2019

Segunda-feira dessa semana

02/09/2019

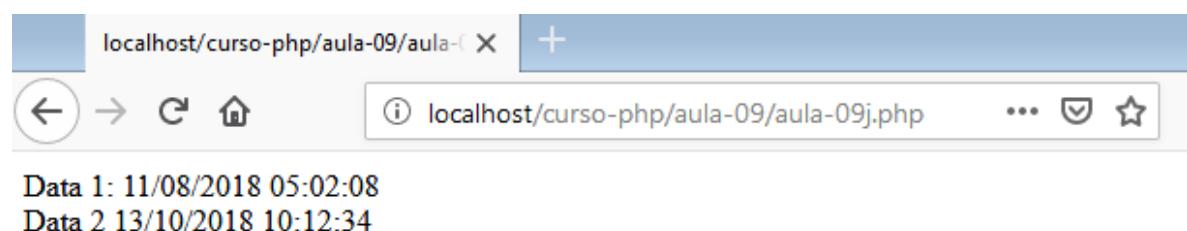
Adicionar 1 mês - 3 dias

02/10/2019

## Classe DateTime (diferença entre datas)

### aula-09/aula-09j.php

```
<?php  
  
$date1 = new DateTime('2018-08-11 05:02:00');  
$date2 = new DateTime('2018-10-13 10:12:34');  
  
$interval = $date1->diff($date2);  
  
echo "Data 1: 11/08/2018 05:02:08 <br>";  
echo "Data 2 13/10/2018 10:12:34 <br><br>";  
  
echo "Diferença entre datas:<br>";  
echo "Meses: {$interval->m}<br> Dias: {$interval->d}<br> Horas: {$interval->h}<br> Minutos:  
{$interval->i}<br> Segundos: {$interval->s}";
```



## DateTime (comparação)

É possível usar operadores de comparação para comparar datas.

### aula-09/aula-09k.php

```
<?php

$hoje = new DateTime('today');
$ontem = new DateTime('yesterday');

echo "hoje > ontem.<br>";
echo "<pre>";
var_dump($hoje > $ontem);
echo "</pre>";
echo "<br>";

echo "hoje < ontem.<br>";
echo "<pre>";
var_dump($hoje < $ontem);
echo "</pre>";
echo "<br>";

echo "hoje == ontem.<br>";
echo "<pre>";
var_dump($hoje == $ontem);
echo "</pre>";
```

localhost/curso-php/aula-09/aula-09k.php

hoje > ontem.  
bool(true)

hoje < ontem.  
bool(false)

hoje == ontem.  
bool(false)

## Typed Properties (PHP 7.4 >=)

Com o advento das typed properties no PHP 7.4, é possível especificar os tipos dos atributos de uma classe e restringir os valores dos mesmos.

O comportamento é similar ao type hint em parâmetros de métodos (funções) e ambos são afetados pelo nível do “strict\_types” onde, o PHP poderá (ou não) converter determinados tipos dos valores em tempo de execução para o tipo determinado na declaração na classe.

### **aula-09/aula-09I.php**

```
<?php

class Veiculo {
    public string $placa;
    public string $modelo;
}

class Pessoa {
    public string $nome;
    public Veiculo $veiculo;
}

$pessoa1 = new Pessoa();
$pessoa1->nome = "Maria da Silva";

$veiculo1 = new Veiculo();
$veiculo1->placa = "ABC 123";

$pessoa1->veiculo = 1; //erro !
$pessoa1->veiculo = $veiculo;
```

## StdClass e Type Cast

A classe nativa do PHP StdClass (Standard Class) tem o propósito de ser uma classe para objetos frutos de coerções. Também permite que objetos genéricos possam ser criados.

### aula-09/aula-09m.php

```
<?php

//convertendo array para objeto
$array = ['nome'=>'José', 'idade'=>34];
$objeto = (object) $array;
echo "<pre>";
print_r($objeto);
echo "</pre>";

//convertendo json para objeto
$json = '{"nome":"Maria", "idade":45}';
$objeto2 = json_decode($json);
echo "<pre>";
print_r($objeto2);
echo "</pre>";

//criando um objeto genérico
$objeto3 = new StdClass();
$objeto3->nome = 'Lucas';
$objeto3->vivo = true;
echo "<pre>";
print_r($objeto3);
echo "</pre>";
```



```
stdClass Object
(
    [nome] => José
    [idade] => 34
)

stdClass Object
(
    [nome] => Maria
    [idade] => 45
)

stdClass Object
(
    [nome] => Lucas
    [vivo] => 1
)
```

# Aula 10 - Form, Upload, \$\_GET, \$\_POST, Session, Cookie

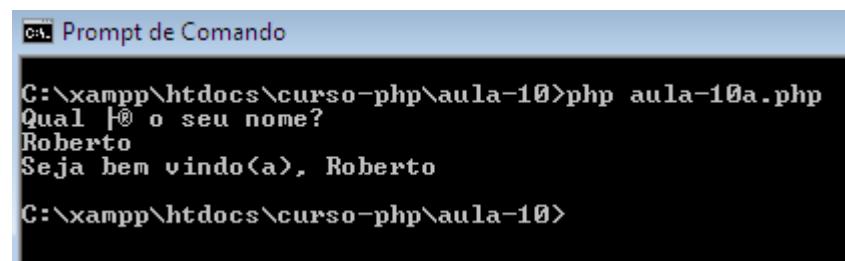
## Entrada de dados

### Entrada de dados

- Para que um aplicação seja dinâmica, faz-se necessário permitir que os usuários possam interagir com a mesma através de algum método de input (entrada) de dados.
- Antes do surgimento dos sistemas operacionais gráficos (GUI), a entrada de dados era feita através de interface de linha de comando (CLI).
- Com o surgimento da web, os dados são, em sua maioria, imputados através de formulários HTML.

### **aula-10/aula-10a.php**

```
<?php  
  
echo "Qual é o seu nome? " . PHP_EOL;  
$input = fgets(STDIN);  
  
echo "Seja bem vindo(a), $input";
```



The screenshot shows a terminal window titled 'Prompt de Comando'. It displays the command 'C:\xampp\htdocs\curso-php\aula-10>php aula-10a.php' being run. The user then types 'Roberto' into the terminal. The output shows the script's response: 'Seja bem vindo(a), Roberto'. The terminal prompt 'C:\xampp\htdocs\curso-php\aula-10>' is visible at the bottom.

```
C:\xampp\htdocs\curso-php\aula-10>php aula-10a.php  
Qual é o seu nome?  
Roberto  
Seja bem vindo(a), Roberto  
C:\xampp\htdocs\curso-php\aula-10>
```

## Entrada de dados web

### aula-10/formulario.html

```
<!DOCTYPE html>

<html>

<head>
    <title>Meu formulário</title>
</head>

<body>

    <form action='cadastrar.php' method='POST'>

        <label>Nome:</label>
        <input type="text" name="nome"/>

        <label>Email:</label>
        <input type="email" name="email"/>

        <input type="submit" value="Cadastrar"/>

    </form>

</body>

</html>
```

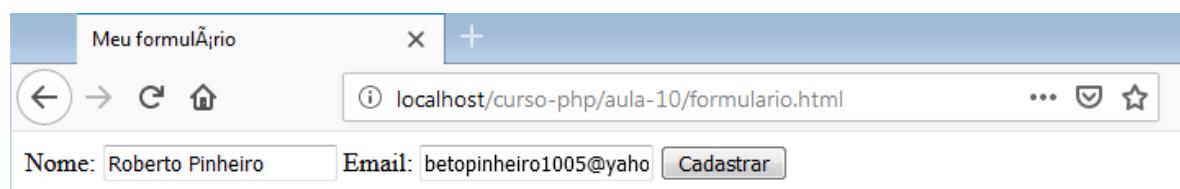
### aula-10/cadastrar.php

```
<?php

echo "<pre>";
print_r($_REQUEST);
echo "</pre>";

echo "Seja bem-vindo " . $_POST['nome'] . "<br>";

?>
```



localhost/curso-php/aula-10/cadas X +

← → ⌂ ⌂

localhost/curso-php/aula-10/cadastrar.php

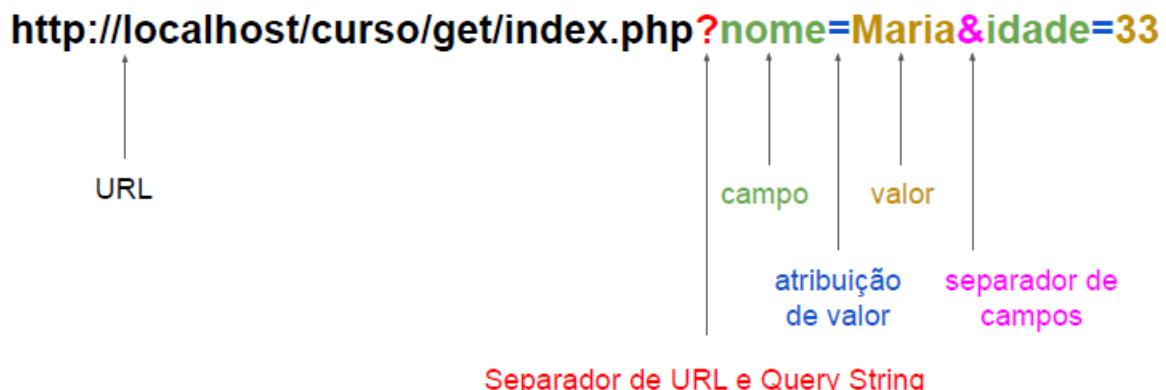
```
Array
(
    [nome] => Roberto Pinheiro
    [email] => betopinheiro1005@yahoo.com.br
)
```

Seja bem-vindo Roberto Pinheiro

## Formulários e Requisições (Variáveis)

- **`$_POST`**: é um array de variáveis com os valores enviados via HTTP POST como, por exemplo, o utilizado por formulários HTML.
- **`$_GET`**: é um array de variáveis com os parâmetros da URL, também conhecido como “query string”.
- **`$_REQUEST`**: por default possui o conteúdo de `$_GET`, `$_POST` (e `$_COOKIE` também).
- **`$_FILES`**: um array com as informações de arquivos enviados via HTTP POST (utilizar o “multipart/form-data” no enctype do formulário)

## Entrada de Dados (GET)



## Formulário (Array como parâmetro)

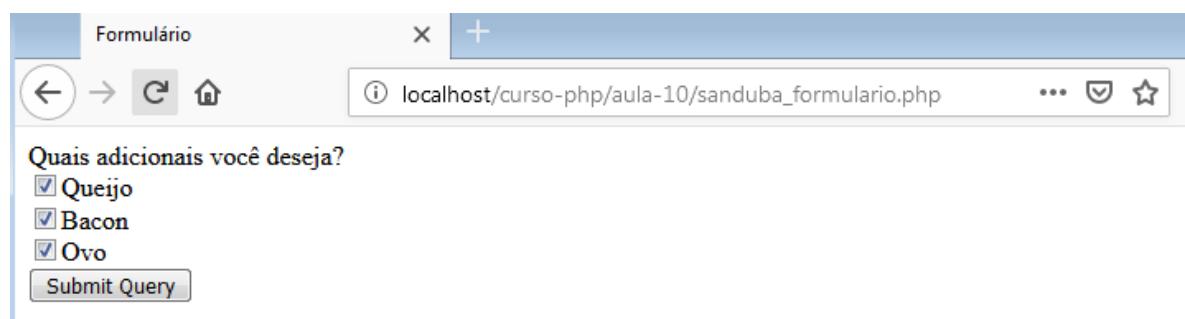
### aula-10/sanduba\_formulario.php

```
<!DOCTYPE html>
<html>
<head>
<title>Formulário</title><meta charset="utf-8" />
</head>
<body>
<form action="sanduba.php" method="post">
Quais adicionais você deseja?<br />
<input type="checkbox" name="adicionais[]" value="queijo" />Queijo<br />
<input type="checkbox" name="adicionais[]" value="bacon" />Bacon<br />
<input type="checkbox" name="adicionais[]" value="ovo" />Ovo<br />
<input type="submit"/>
</form>
</body>
</html>
```

### aula-10/sanduba.php

```
<?php

foreach($_REQUEST['adicionais'] as $adicional){
    echo $adicional."<br/>";
}
```

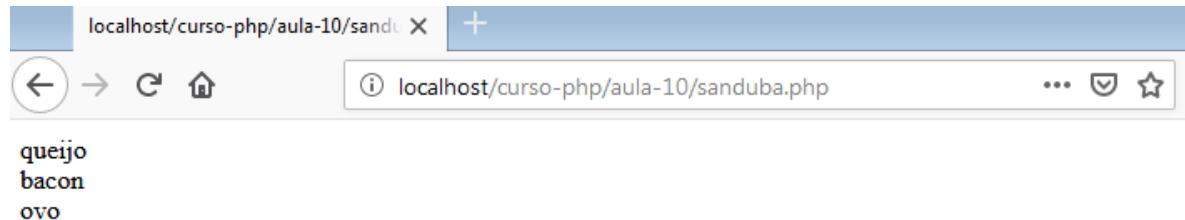


Formulário

Quais adicionais você deseja?

Queijo  
 Bacon  
 Ovo

Submit Query



localhost/curso-php/aula-10/sanduba.php

localhost/curso-php/aula-10/sanduba.php

queijo  
bacon  
ovo

## Formulários (upload - `$_FILES` e `multipart/form-data`)

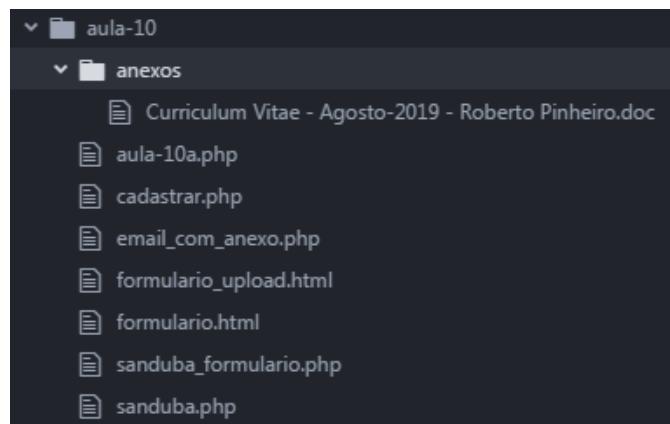
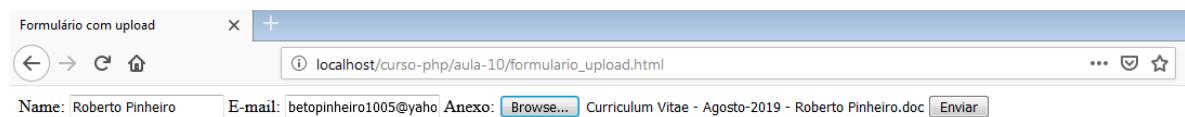
### aula-10/formulario\_upload.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Formulário com upload</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <form action="email_com_anexo.php"
      method="post" enctype="multipart/form-data">
      Name: <input type="text" name="nome">
      E-mail: <input type="email" name="email">
      Anexo: <input type="file" name="anexo">
      <input type="submit" value="Enviar"/>
    </form>
  </body>
</html>
```

### aula-10/email\_com\_anexo.php

```
<?php
$diretorio = "anexos".DIRECTORY_SEPARATOR;

foreach($_FILES as $arquivo){
  $nome = $arquivo['name'];
  $conteudo = file_get_contents($arquivo['tmp_name']);
  //salvando no disco
  file_put_contents($diretorio.$nome, $conteudo);
}
```



## Cookie

Persiste dados do lado do cliente.

Um cookie é um pequeno arquivo que o servidor incorpora no computador (navegador) do usuário. Cada vez que o mesmo computador solicita uma página com um navegador, ele também enviará o cookie.

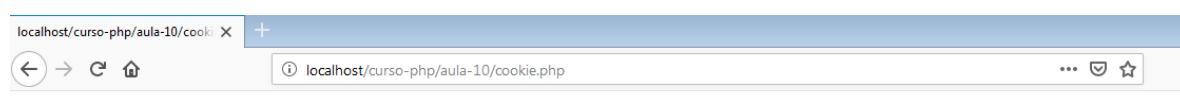
Com o PHP, é possível criar e recuperar valores de cookie. O nome do cookie é automaticamente atribuído a uma variável com o mesmo nome.

A função `setcookie()` cria o cookie enquanto a variável global `$_COOKIE` acessa os valores.

O cookie pode ser substituído por soluções mais modernas como o localStorage do Javascript.

### aula-10/cookie.php

```
<?php  
  
$cookie_name = "usuario";  
$cookie_value = "João da Silva";  
setcookie($cookie_name, $cookie_value, mtime(24), "/");  
  
?>
```

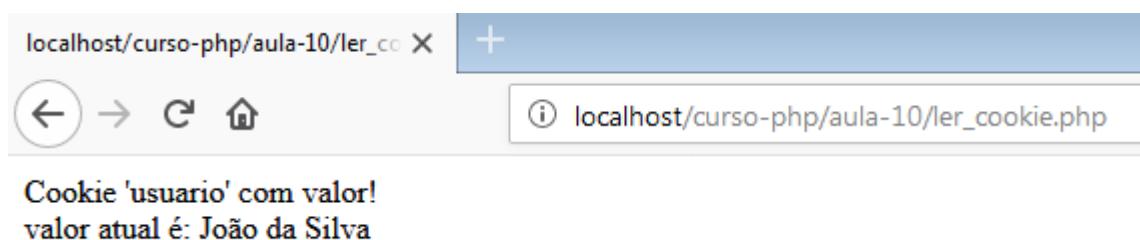


Name	Domain	Path	Expires on	Last accessed on	Value
usuario	localhost	/	Fri, 06 Sep 2019 22:07:54 GMT	Fri, 06 Sep 2019 02:07:54 GMT	Jo%C3%A3o+da+Silva

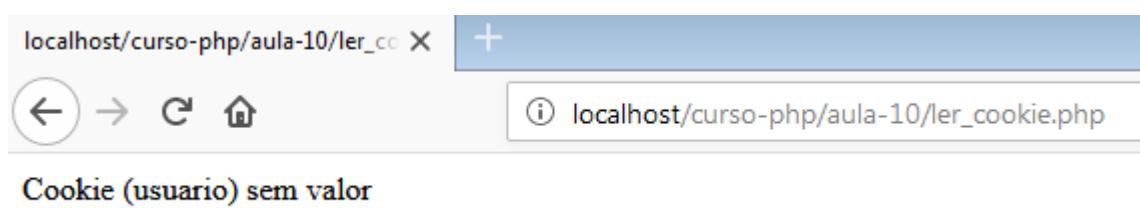
Name	Domain	Path	Expires on	Last accessed on	Value
usuario	localhost	/	Fri, 06 Sep 2019 22:07:54 GMT	Fri, 06 Sep 2019 02:07:54 GMT	Jo%C3%A3o+da+Silva

## **aula-10/ler\_cookie.php**

```
<!DOCTYPE html>
<html>
<body>
<?php
if (!isset($_COOKIE[$cookie_name])) {
    echo "Cookie (" . $cookie_name . ") sem valor";
} else {
    echo "Cookie '" . $cookie_name . "' com valor!<br>";
    echo "valor atual é: " . $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```



Removendo manualmente o cookie:



## Session

Persiste dados do lado do servidor.

Session (sessão) é uma maneira de armazenar informações temporariamente que podem ser utilizadas em várias páginas da aplicação.

Ao contrário de um cookie, as informações não são armazenadas no computador do usuário e sim no servidor.

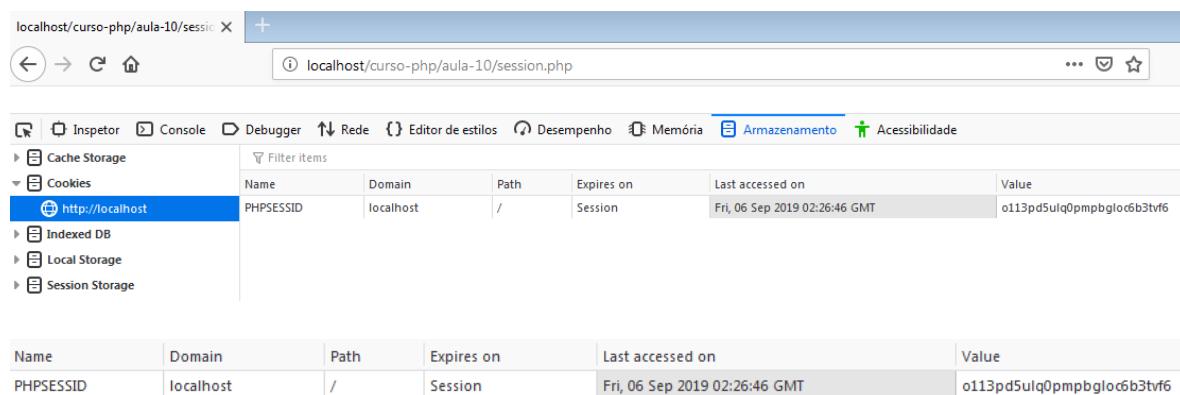
Apesar da informação (dado) estar no servidor, um pequeno cookie é criado no navegador do usuário para vincular a este dado(s) na sessão.

É possível destruir a sessão utilizando a função: session\_destroy();

### aula-10/session.php

```
<?php session_start(); ?>

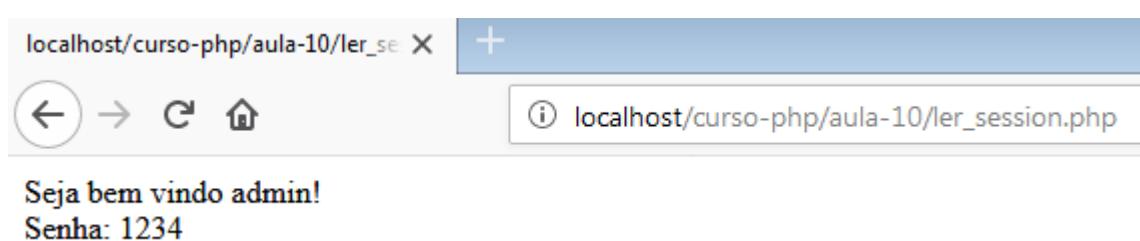
<!DOCTYPE html>
<html>
  <body>
    <?php
      $_SESSION["usuario"] = "admin";
      $_SESSION["senha"] = "1234";
    ?>
  </body>
</html>
```



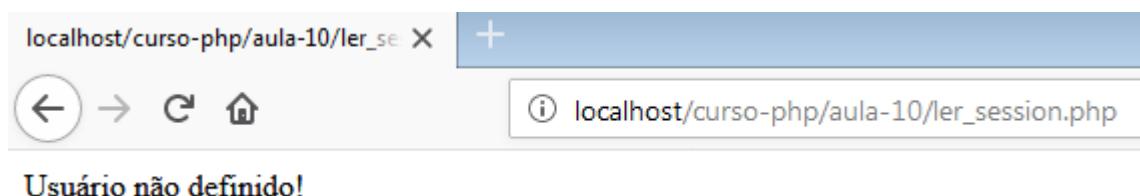
The screenshot shows the browser developer tools Network tab. The address bar indicates the URL is `localhost/curso-php/aula-10/session.php`. The Network tab is selected, and the Cookies section is expanded, showing a single cookie named `PHPSESSID` with the value `o113pd5ulq0pmpbglc6b3tvf6`. The cookie is set for the domain `localhost` and has a session expiration. Other storage types like Cache Storage, Indexed DB, and Local Storage are also listed in the sidebar.

## **aula-10/ler\_session.php**

```
<?php  
  
session_start();  
  
if(isset($_SESSION['usuario'])){  
    echo "Seja bem vindo {$_SESSION['usuario']}!<br>";  
    echo "Senha: {$_SESSION['senha']}";  
}  
else{  
    // header("Location: index.php");  
    echo "Usuário não definido!";  
}
```



Removendo a session (do servidor):



# Aula 11 - Ajax, Fetch, Formdata, Upload múltiplo, FetchJson

## PHP Assíncrono (PHP + Javascript)

- Com o fluxo tradicional “síncrono” do HTTP, faz-se necessário “atualizar/carregar” a página toda vez que um informação é enviada do cliente para o servidor (ou o inverso), o que, em muitas situações, pode prejudicar a interação do usuário com a aplicação.
- Com o advento do Ajax em 1999, os apps web tornaram-se capaz de enviar e recuperar dados do servidor de forma assíncrona (em segundo plano) sem interferir na exibição e no comportamento da página existente. Ao separar a camada de troca de dados da camada de apresentação, o Ajax permite que uma página web possa alterar seu conteúdo dinamicamente sem a necessidade de recarregar a página inteira.
- A partir de 2017, uma nova alternativa surgiu ao Ajax: o fetch, que permite que requisições assíncronas possam ser feitas usando o conceito de promises.

Com o uso do XMLHttpRequest do Javascript é possível recuperar enviar dados e receber conteúdos de documentos web de forma assíncrona.

## PHP Assíncrono (Ajax - XMLHttpRequest)

### aula-11/ajax/index.php

```
<!DOCTYPE html>

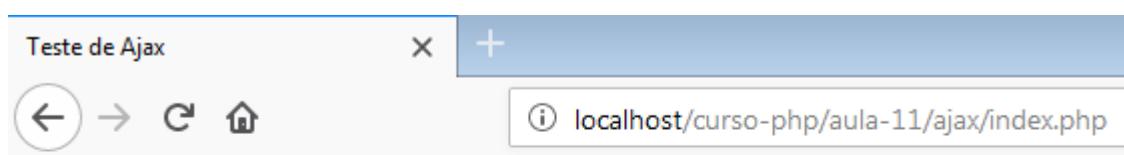
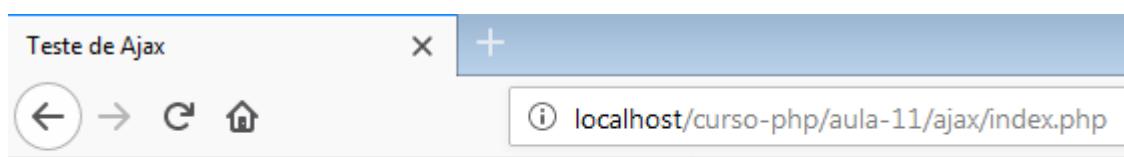
<html lang="pt-BR">
  <head>
    <title>Teste de Ajax</title>
    <script src="curso.js"></script>
  </head>
  <body>
    <div id="mensagem"></div>
    <br>
    <button onclick="meuAjax()">Ver mensagem</button>
  </body>
</html>
```

## **aula-11/ajax/curso.js**

```
function meuAjax() {  
    var xhr = new XMLHttpRequest();  
    xhr.open('GET', 'meu_ajax.php?nome=Maria');  
    xhr.onload = function () {  
        if (xhr.status === 200) {  
            document.getElementById('mensagem')  
                .innerHTML = xhr.responseText;  
        } else {  
            alert('Erro! Status: ' + xhr.status);  
        }  
    };  
  
    xhr.send();  
}  
}
```

## **aula-11/ajax/meu\_ajax.php**

```
<?php  
  
echo "Meu nome é ". $_REQUEST['nome'] . "<br>";  
echo " " . (new DateTime())->format('h:i:s');
```





Network										
Status	Method	Domain	File	Cause	Type	Transferred	Size	0 ms	20,48 s	40,96 s
All										
200	GET	localhost	curso.js	script	js	657 B	336 B	4 ms		
404	GET	localhost	favicon.ico	img	html	cached	209 B			
200	GET	localhost	index.php	document	html	513 B	259 B	67 ms		
200	GET	localhost	meu_ajax.php?nome=Maria	xhr	html	283 B	30 B			244 ms

Headers	Cookies	Params	Response	Timings	Stack Trace
Preview					

Meu nome é Maria  
07:23:01

## PHP Assíncrono (Fetch JS)

A partir de 2017, o fetch permite que requisições assíncronas possam ser feitas usando o conceito de promises.

### aula-11/fetch/index.php

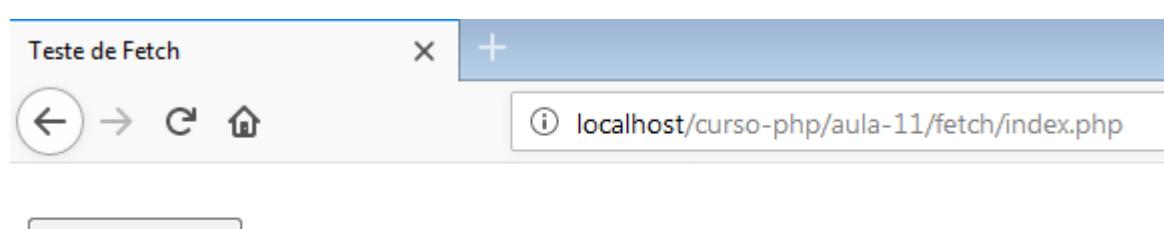
```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>Teste de Fetch</title>
    <script src="curso.js"></script>
  </head>
  <body>
    <div id="mensagem"></div>
    <br>
    <button onclick="meuFetch()">Ver mensagem</button>
  </body>
</html>
```

### aula-11/fetch/curso.js

```
function meuFetch() {
  window.fetch("meu_fetch.php?nome=Maria")
  .then(response => response.text())
  .then(data => {
    document.getElementById('mensagem')
    .innerHTML = data;
  })
  .catch(error => alert('Erro!' + error));
}
```

### aula-11/fetch/meu\_fetch.php

```
<?php
echo "Meu nome é ".$_REQUEST['nome']."<br>";
echo " ".(new DateTime())->format('h:i:s');
```



Teste de Fetch

localhost/curso-php/aula-11/fetch/index.php

Meu nome é Maria  
07:34:35

[Ver mensagem](#)

Inspecor

Search HTML

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head></head>
  <body>
    <div id="mensagem">
      Meu nome é Maria
      <br>
      07:34:35
      <br>
    </div>
    <br>
    <button onclick="meuFetch()">Ver mensagem</button> event
  </body>
</html>
```

Inspecor

File

Status	Met...	Domain	File	Cause	Type	Transferred	Size	0 ms	20,48 s	40,96 s
304	GET	localhost	curso.js	script	js	cached	242 B	2 ms		
404	GET	localhost	favicon.ico	img	html	cached	209 B			
200	GET	localhost	index.php	document	html	513 B	259 B	265 ms		
200	GET	localhost	meu_fetch.php?nome=Maria	fetch	html	287 B	34 B			33 ms

Headers Cookies Params Response Timings Stack Trace

Preview

Meu nome é Maria  
07:34:35

## PHP Assíncrono (FormData + Fetch JS)

### aula-11/form\_data/index.php

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>Formulário Assíncrono</title>
    <script src="curso.js"></script>
  </head>
  <body>
    <form id="meu_form" action="meu_form_data.php">
      <label>Nome:</label>
      <input type="text" name="nome"/>
      <label>Email:</label>
      <input type="email" name="email"/>
      <input type="submit" value="Cadastrar"/>
    </form>
    <div id="mensagem"></div>
    <script>
      const form = document.getElementById('meu_form');
      form.addEventListener('submit', meuFormData);
    </script>
  </body>
</html>
```

### aula-11/form\_data/curso.js

```
function meuFormData(event) {
  event.preventDefault();
  const formData = new FormData(this);
  window.fetch(this.getAttribute("action"), {
    method: 'post',
    body: formData
  }).then(function (response) {
    return response.text();
  }).then(function (text) {
    document.getElementById('mensagem')
      .innerHTML = text;
  });
}
```

### aula-11/form\_data/meu\_form\_data.php

```
<?php
echo "Email {$_REQUEST['email']} cadastrado com sucesso !";
```

Formulário Assíncrono +

localhost/curso-php/aula-11/form\_data/index.php

**Nome:**  **Email:**  **Cadastrar**

Formulário Assíncrono +

localhost/curso-php/aula-11/form\_data/index.php

**Nome:**  **Email:**  **Cadastrar**

Email **betopinheiro1005@yahoo.com.br** cadastrado com sucesso !

✖ Inspecor Console Debugger ⟳ Rede {} Editor de estilos ⌚ Desempenho ⌚ Memória 🗄 Armazenamento 👤

✖ Filter URLs

Status	Met...	Domain	File	Cause	Type	Transferred	Size	0 ms	20,48 s	40,96 s
200	GET	localhost	curso.js		script	js	673 B	352 B	3 ms	
404	GET	localhost	favicon.ico		img	html	cached	209 B		
200	GET	localhost	index.php		document	html	852 B	598 B	130 ms	
200	POST	localhost	meu_form_data.php		fetch	html	313 B	60 B		352 ms

✖ Headers Cookies Params Response Timings Stack Trace

▼ Preview

Email **betopinheiro1005@yahoo.com.br** cadastrado com sucesso !

## PHP Assíncrono (Upload Múltiplo com Fetch)

### aula-11/upload\_multiplo/index.php

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head> <title>Formulário Assíncrono</title>
    <script src="curso.js"></script> </head>
  <body>
    <form id="meu_form" action="meu_form_data.php" enctype="multipart/form-data">
      <label>Nome:</label>
      <input type="text" name="nome"/>
      <label>Email:</label>
      <input type="email" name="email"/>
      <input type="file" name="anexo" multiple="multiple"/>
      <input type="submit" value="Cadastrar"/>
    </form>
    <div id="mensagem"></div>
    <script>
      const form = document.getElementById('meu_form');
      form.addEventListener('submit', meuFormData);
    </script>
  </body>
</html>
```

### aula-11/upload\_multiplo/curso.js

```
function meuFormData(event) {
  event.preventDefault();
  const formData = new FormData(this);
  var fileInput = document.querySelector('input[type="file"]');
  formData.delete('anexo');
  for (var i=0; i < fileInput.files.length; i++){
    formData.append('anexo'+i, fileInput.files.item(i));
  }
  window.fetch(this.getAttribute("action"), {
    method: 'post',
    body: formData
  }).then(function (response) {
    return response.text();
  }).then(function (text) {
    document.getElementById('mensagem').innerHTML = text;
  });
}
```

## **aula-11/upload\_multiplo/meu\_form\_data.php**

```
<?php  
echo "Email {$_REQUEST['email']} cadastrado com sucesso ! <br/>";  
  
$diretorio = "anexos".DIRECTORY_SEPARATOR;  
  
foreach($_FILES as $arquivo){  
    $nome = $arquivo['name'];  
    $conteudo = file_get_contents($arquivo['tmp_name']);  
    file_put_contents($diretorio.$nome, $conteudo);  
    if(file_exists($diretorio.$nome))  
        echo "Arquivo {$arquivo['name']} salvo com sucesso ! <br/>";  
}
```

Formulário Assíncrono    +

← → C ⌂

localhost/curso-php/aula-11/upload\_multiplo/index.php

Nome: Roberto Pinheiro Email: betopinheiro1005@yahoo.com.br Browse... 3 arquivos selecionados. Cadastrar

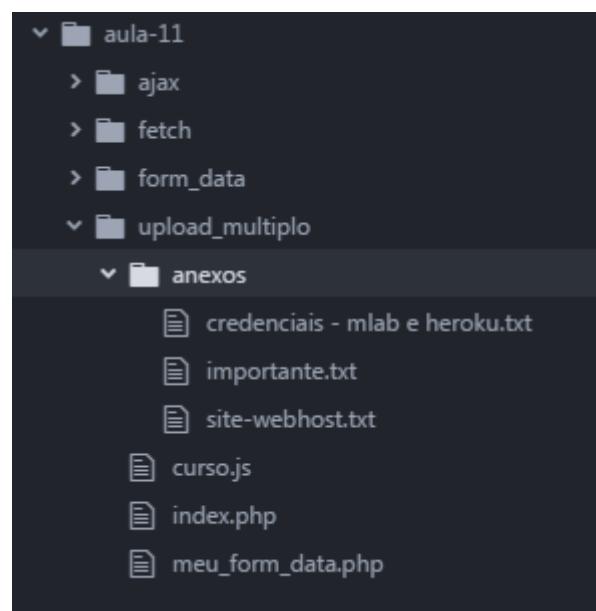
Formulário Assíncrono    +

← → C ⌂

localhost/curso-php/aula-11/upload\_multiplo/index.php

Nome: Roberto Pinheiro Email: betopinheiro1005@yahoo.com.br Browse... 3 arquivos selecionados. Cadastrar

Email betopinheiro1005@yahoo.com.br cadastrado com sucesso !  
Arquivo credenciais - mlab e heroku.txt salvo com sucesso !  
Arquivo importante.txt salvo com sucesso !  
Arquivo site-webhost.txt salvo com sucesso !



Inspetor Console Debugger Editor de estilos Desempenho Memória Armazenamento

Filter URLs

Status	Met...	Domain	File	Cause	Type	Transferred	Size	0 ms		40,96 s
304	GET	localhost	curso.js		script	js	cached	554 B	1 ms	
404	GET	localhost	favicon.ico		img	html	cached	209 B		
200	GET	localhost	index.php		document	html		935 B	269 ms	
200	POST	localhost	meu_form_data.php		fetch	html		483 B	229 B	254 ms

Headers Cookies Params Response Timings Stack Trace

▼ Preview

Email betopinheiro1005@yahoo.com.br cadastrado com sucesso !

Arquivo credenciais - mlab e heroku.txt salvo com sucesso !

Arquivo importante.txt salvo com sucesso !

Arquivo site-webhost.txt salvo com sucesso !

## PHP Assíncrono (JSON + <select>)

### aula-11/json\_select/index.php

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>Formulário Assíncrono</title> <script src="curso.js"></script>
  </head>
  <body>
    <form>
      <label>Região:</label>
      <select id="regioes">
        <option value="">Selecione...</option>
        <option value="centro-oeste">Centro Oeste</option>
        <option value="sul">Sul</option>
      </select>
      <label>Estado:</label>
      <select id="estados"></select>
    </form>
    <script>
      const select = document.getElementById('regioes');
      select.addEventListener('change', selectEstados.bind(this,'regioes', 'estados'), false);
    </script>
  </body>
</html>
```

### aula-11/json\_select/curso.js

```
function selectEstados(fonte_id, alvo_id) {
  fonte = document.getElementById(fonte_id);
  alvo = document.getElementById(alvo_id);
  alvo.length = 0;
  let regiao_selecionada = fonte.options[fonte.selectedIndex].value;
  if (regiao_selecionada == "")
    return;
  window.fetch("estados.php?regiao_selecionada=" + regiao_selecionada)
    .then(response => response.json())
    .then(data => {
      for (var i = 0; i < data.length; i++) {
        var option = document.createElement("option");
        option.innerHTML = data[i].nome;
        option.value = data[i].id;
        alvo.options.add(option);
      }
    })
    .catch(error => alert('Erro!' + error));
}
```

## **aula-11/json\_select/estados.php**

```
<?php
header('Content-Type: application/json');
$regiao_selecionada = $_REQUEST['regiao_selecionada'];
$estados = [
    'centro-oeste' => [
        ['id'=>'DF', 'nome' => 'Distrito Federal'],
        ['id'=>'GO', 'nome' => 'Goiás'],
        ['id'=>'MT', 'nome' => 'Mato Grosso'],
        ['id'=>'MS', 'nome' => 'Mato Grosso do Sul']
    ],
    'sul' => [
        ['id'=>'PR', 'nome'=> 'Paraná'],
        ['id'=>'RS', 'nome'=> 'Rio Grande do Sul'],
        ['id'=>'SC', 'nome'=> 'Santa Catarina']
    ]
];
echo json_encode($estados[$regiao_selecionada]);
```

Formulário Assíncrono +

← → ↻ ⌂ localhost/curso-php/aula-11/json\_select/

Região:  Estado:

Formulário Assíncrono +

← → ↻ ⌂ localhost/curso-php/aula-11/json\_select/

Região:  Estado:   
Paraná  
Rio Grande do Sul  
Santa Catarina

Formulário Assíncrono +

← → ↻ ⌂ localhost/curso-php/aula-11/json\_select/

Região:  Estado:   
Distrito Federal  
Goiás  
Mato Grosso  
Mato Grosso do Sul

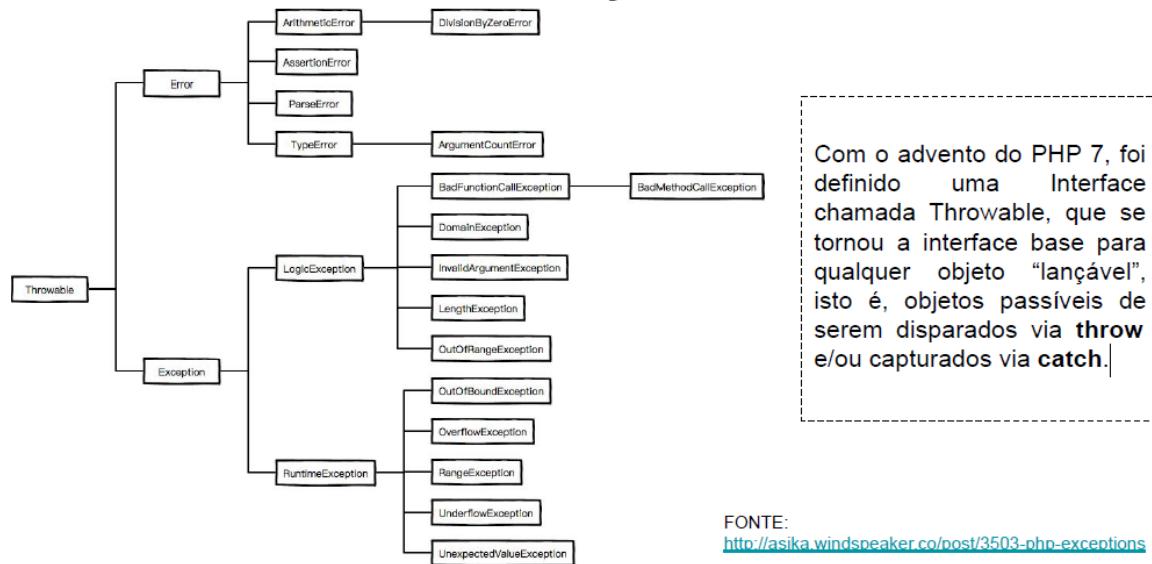
# Aula 12 - Throwables, Exceptions, Erros e Serialização

## Tratamento de Exceções/Erros

- O tratamento de exceção é usada para alterar o fluxo normal da execução de código se ocorrer uma condição de erro específico (especial). Essa condição é chamada de exceção.
- No PHP 7>= existem dois principais tipos de erros: os throwables e o não throwables.
- Os throwables podem ser lançados via throw e tratados via try/catch. Nesta categoria existem dois sub tipos: Exceptions e Errors.
- Os não throwables podem, em alguns casos, ser suprimidos via @, error\_reporting ou tratados utilizando o set\_error\_handler()

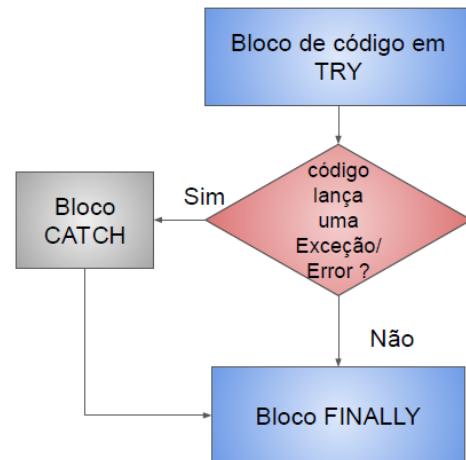
Com o advento do PHP 7, foi definido uma Interface chamada Throwable, que se tornou a interface base para qualquer objeto “lançável”, isto é, objetos passíveis de serem disparados via throw e/ou capturados via catch.

## Taxonomia de Erros e Exceções (throwables)



# Manipulação de Throwables

```
<?php  
  
try {  
    // código aqui  
}  
catch (\Throwable $t) {  
    echo $t->getMessage();  
}
```

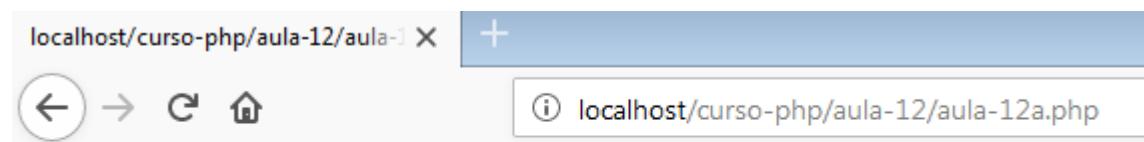


## Throwable

Utilizando a Interface base como typehint do catch, é possível capturar tanto as Exceptions como os Errors.

### aula-12/aula12-a.php

```
<?php  
  
try {  
    $resultado = intdiv(10,0);  
    echo "Resultado: ". $resultado . "<br>";  
} catch(\Throwable $t) {  
    echo $t->getMessage() . "<br>";  
} finally {  
    echo "Terminou o nosso try!";  
}
```

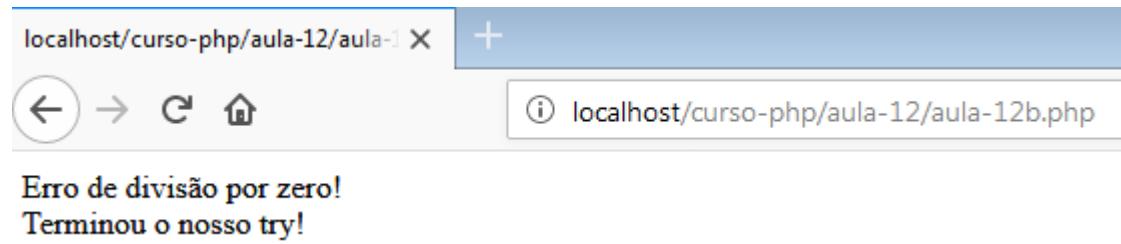


## Tratamento de Throwables (múltiplos)

É possível especificar diferentes tratamentos de Exceptions/Errors através do aninhamento de catchs.

### aula-12/aula12-b.php

```
<?php
try {
    $resultado = intdiv(10,0);
    echo "Resultado: ". $resultado . "<br>";
} catch(DivisionByZeroError $t) {
    echo "Erro de divisão por zero!<br>";
} catch(TypeError $t) {
    echo "Erro de tipagem!<br>";
} finally {
    echo "Terminou o nosso try!";
}
```

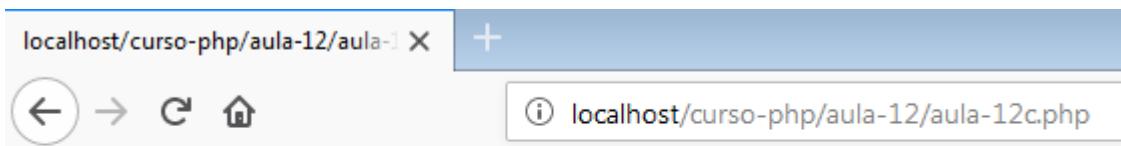


## Tratamento de Throwables (grupo com pipe)

Com o advento do PHP 7.1 é possível especificar um único tratamento para um grupo de diferentes tratamentos de Exceptions/Errors utilizando o operador pipe "|";

### aula-12/aula12-c.php

```
<?php
try {
    $resultado = intdiv(10,0);
    echo "Resultado: ". $resultado . "<br>";
} catch(DivisionByZeroError | TypeError $t) {
    echo "Erro de divisão por zero ou erro de tipagem!<br>";
} finally {
    echo "Terminou o nosso try!";
}
```



## Lançamento de Throwables (throw new)

Com o uso do operador `throw` seguido de uma instância de `Throwable` é possível “lançar” uma exceção/erro para que o usuário de uma função/método possa receber as informações necessárias do erro e tratá-lo da melhor maneira possível;

Se o desenvolvedor tentar usar uma instância de um objeto que não implementa `Throwable` ocorrerá um erro fatal do tipo “`Uncaught Error`” descrevendo que o objeto não implementa a interface;

### **aula-12/aula12-d.php**

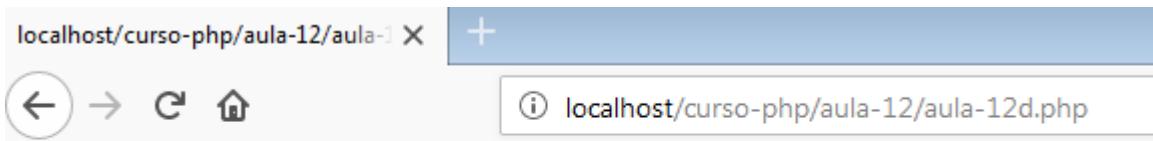
```
<?php

function escolhaMcOferta (int $opcao) : ?string {

    $ofertas = ['bic mac', 'mc cheddar',
    'quarteirão', 'mc fish', 'mc chicken'];

    if (!in_array($opcao, range(1,5))) {
        throw new OutOfRangeException("Oferta inválida!");
    }
    return $ofertas[--$opcao];
}

try {
    echo escolhaMcOferta(6);
} catch (\Throwable $t) {
    echo "Ocorreu um erro: " . $t->getMessage();
}
```



Ocorreu um erro: Oferta inválida!

## Criando Exceções/Erros customizados

É possível criar um Throwable customizado com uma especialização de Exception ou Error. (Pois não é possível implementar a interface Throwable).

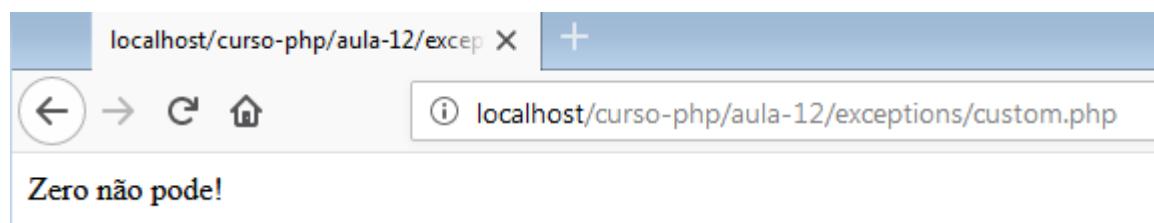
### aula-12/exceptions/custom.php

```
<?php

class MeuException extends Exception {
    public function __construct($message = ""){
        parent::__construct($message);
        file_put_contents('log.txt', (new DateTime())->format('d/m/Y') . $this->getTraceAsString()."<br>" ,
FILE_APPEND | LOCK_EX);
    }
}

function testar($x){
    if($x == 0){
        throw new MeuException('Zero não pode!');
    }
}

try{
    testar(0);
}catch(Exception $e){
    echo $e -> getMessage();
}
```



### aula-12/exceptions/log.txt

```
07/09/2019#0 C:\xampp\htdocs\curso-php\aula-12\exceptions\custom.php(18):
testar(0)
#1 {main}<br>
```

## Principais Tipos de Erros (não Throwables)

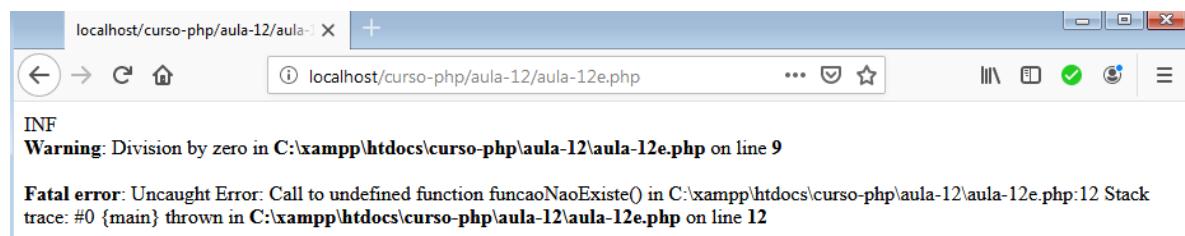
Tipo	Constante	Significado	Interrompe a execução do script?	Pode ser suprimido com @?	Tempo de:
NOTICE	E_NOTICE	Aviso para indicar que o script encontrou alguma coisa que pode indicar um erro;	NÃO	SIM	Execução
WARNING	E_WARNING	Erro não fatal;	NÃO	SIM	Execução
DEPRECATED	E_DEPRECATED	Aviso de um recurso depreciado e será futuramente removido nas próximas versões do PHP.	NÃO	SIM	Execução
ERROR	E_ERROR	Erro fatal em tempo de execução. Estes indicam erros que não podem ser recuperados.	SIM	NÃO	Execução
PARSER ERROR	E_PARSER	Erros gerados pelo interpretador devido a erro de sintaxe no script	SIM	NÃO	Compilação

## Operador de controle de erro (supressão @)

O PHP suporta um operador de controle de erro: o sinal 'arroba' (@). Com uso do @ é possível suprimir um notice ou um erro não fatal.

### aula-12/aula12-e.php

```
<?php
echo @(10 / 0);
// supriu "Warning: Division by zero"
$c = @$_POST["nome"] . @$_POST["sobrenome"];
// supriu "Notice: Undefined index: nome"
// supriu "Notice: Undefined index: sobrenome"
@$newfunc = create_function('$a', 'return;');
// supriu "Deprecated: Function create_function() is deprecated"
@ $i / 0;
// supriu "Notice: Undefined variable: i"
// não supriu o "Warning: Division by zero"
$c = @funcaoNaoExiste(); //não supriu erro fatal
echo 'fim';
```



## Controle de nível de erros não throwables

Com o uso da função error\_reporting() podemos controlar, em tempo de execução quais avisos de erros o PHP poderá imprimir na tela. Essas mesmas constantes podem ser setadas de uma maneira global no php.ini.

### **aula-12/aula12-f.php**

```
<?php
// Desligando todos os avisos de erros
error_reporting(0);
// Ligando apenas para warning
error_reporting(E_WARNING);
// com o uso do pipe | é possível criar uma lista
// fixa de avisos para erros
error_reporting(E_ERROR | E_WARNING | E_PARSE);
// E_ALL é o equivalente a todos os tipos
error_reporting(E_ALL);
// Com o uso do ^ (not) é possível remover
// um item da lista (todos menos notice)
error_reporting(E_ALL ^ E_NOTICE);
```

## **set\_error\_handler() e restore\_error\_handler()**

Com o uso do set\_error\_handler() é possível criar um manipulador customizado de erros não throwables. É possível, inclusive, lançar uma exceção dentro de um manipulador customizado permitindo assim que erro possa ser tratado com um throwable, podendo ser capturado via catch.

O ideal é sempre no final do bloco de instruções restaurar o manipulador original do programa com o restore\_error\_handler().

## **aula-12/aula12-g.php**

```
<?php

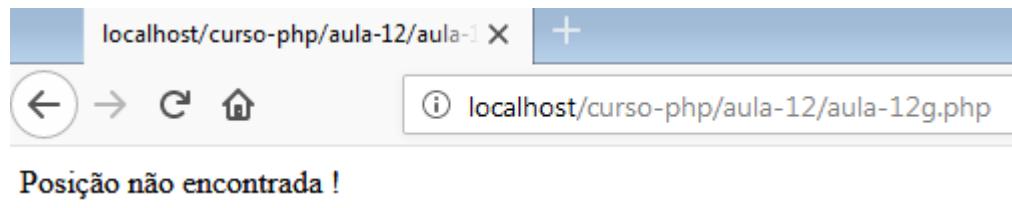
set_error_handler("manipuladorCustomizadoDeErros");

function manipuladorCustomizadoDeErros
    ($severity, $mensagem, $arquivo, $linha) {
        if (error_reporting() & $severity) {
            throw new Exception($mensagem, 0);
        }
    }

$array = ['maria','jósé'];

try {
    $b = $array[2];
} catch (Exception $e) {
    echo "Posição não encontrada !";
} finally {

    restore_error_handler();
}
}
```



## **Serialização**

### **Serialização**

- É o processo de traduzir estruturas de dados ou estado de objeto em um formato que pode ser armazenado (por exemplo, em um arquivo ou buffer de memória) ou transmitido e reconstruído posteriormente (possivelmente em um ambiente de computador diferente).
- Quando a série resultante de bits é relida de acordo com o formato de serialização, ela pode ser usada para criar um clone semanticamente idêntico do objeto original.
- Esse processo de serializar um objeto também é chamado de **marshalling** (empacotamento de um objeto).
- A operação oposta, extraíndo uma estrutura de dados de uma série de bytes, é a desserialização (também chamada de **unmarshalling**).

## **aula-12/serializacao/Aluno.php**

```
<?php

class Aluno {
    public $nome;
    public $matricula;
    public function __construct(string $nome, int $matricula) {
        $this->nome = $nome;
        $this->matricula = $matricula;
    }
}
```

## **aula-12/serializacao/Turma.php**

```
<?php

class Turma {
    public $nome;
    public $data;
    public $alunos;

    public function __construct(string
        $nome, \DateTime $data, array $alunos = []) {
        $this->nome = $nome;
        $this->data = $data;
        $this->alunos = $alunos;
    }
}
```

## Serialização (escrita)

A função `serialize` converte toda estrutura de arrays e objetos em uma string utilizando um formato passível de recuperação pelo próprio PHP. Já a função `file_put_contents` persiste uma string em um arquivo texto. Se as classes não forem carregadas o PHP irá gerar instâncias de `__PHP_Incomplete_Class`.

### **aula-12/serializacao/index.php**

```
<?php

include_once 'Turma.php';
include_once 'Aluno.php';

$turmas = [];

$aluno1 = new Aluno('José', 123);
$aluno2 = new Aluno('Maria', 456);
$aluno3 = new Aluno('Thiago', 789);

$turmas[] = new Turma('PHP',
    new \DateTime('today'), [$aluno1, $aluno2]);
$turmas[] = new Turma('CakePHP',
    new \DateTime('-2 days'), [$aluno1, $aluno2, $aluno3]);
$turmas[] = new Turma('MySQL',
    new \DateTime('yesterday'), [$aluno1, $aluno3]);
$serializacao = serialize($turmas);

file_put_contents('dados.db', $serializacao);
```

Ao rodar, gera o arquivo dados.db

### **aula-12/serializacao/dados.db**

```
a:3:{i:0;O:5:"Turma":3:{s:4:"nome";s:3:"PHP";s:4:"data";O:8:"DateTime"
:s:4:"date";s:26:"2019-09-08
00:00:00.000000";s:13:"timezone_type";i:3;s:8:"timezone";s:13:"Europe/
Berlin";}s:6:"alunos";a:2:{i:0;O:5:"Aluno":2:{s:4:"nome";s:5:"José";s:9:"matricula";i:123;}i:1;O:5:"Aluno":2:{s:4:"nome";s:5:"Maria";s:9:"matricula";i:456;}}i:1;O:5:"Turma":3:{s:4:"nome";s:7:"CakePHP";s:4:"data";O:8
:"DateTime":3:{s:4:"date";s:26:"2019-09-06
01:27:13.518147";s:13:"timezone_type";i:3;s:8:"timezone";s:13:"Europe/
Berlin";}s:6:"alunos";a:3:{i:0;r:9;i:1;r:12;i:2;O:5:"Aluno":2:{s:4:"nome";
s:6:"Thiago";s:9:"matricula";i:789;}}i:2;O:5:"Turma":3:{s:4:"nome";s:5:
"MySQL";s:4:"data";O:8:"DateTime":3:{s:4:"date";s:26:"2019-09-07
00:00:00.000000";s:13:"timezone_type";i:3;s:8:"timezone";s:13:"Europe/
Berlin";}s:6:"alunos";a:2:{i:0;r:9;i:1;r:24;}}}}
```

### **Desserialização (leitura)**

A função `file_get_contents` recupera o conteúdo de um arquivo texto em uma string. A função `unserialize` interpreta o conteúdo serializado em uma string e reconstrói os objetos e arrays.

### **aula-12/serializacao/unserialize.php**

```
<?php

include 'Turma.php';
include 'Aluno.php';

$serializacao = file_get_contents('dados.db');
$turmas = unserialize($serializacao);

echo "<table border>";
foreach ($turmas as $turma) {
    echo "<tr>";
    echo "<td> {$turma->nome} </td>";
    echo "<td> {$turma->data->format('d/m/Y')} </td>";
    echo "<td>". implode(", ", array_column($turma->alunos, 'nome'))."</td>";
    echo "</tr>";
}
echo "</table>";
```

localhost/curso-php/aula-12/serializacao X +

← → ⌂ ⌂

localhost/curso-php/aula-12/serializacao/leitura.php

PHP	08/09/2019	José, Maria
CakePHP	06/09/2019	José, Maria, Thiago
MySQL	07/09/2019	José, Thiago

# Aula 13 - MySQL, Postgres, SQLServer, Oracle com PDO - Parte 1

## PHP e Bancos de Dados Relacionais

- Até o PHP 5.0, era necessário utilizar extensões PECL com conjuntos diferentes de funções para cada acessar cada banco de dados (ex.: `mysql_connect`, `mssql_connect` etc...).
- Com o advento do PHP 5.1 surgiu o PDO (*PHP Data Objects*): uma camada de abstração de acesso a banco de dados (DBAL). Uma DBAL (Database abstraction layer) é uma API que visa unificar a comunicação entre um aplicação e bancos de dados.
- As DBALs reduzem a quantidade de trabalho para acessar diferentes bancos de dados fornecendo uma API consistente ao desenvolvedor ocultando o máximo possível as especificidades do banco de dados por trás dessa interface.
- O PDO fornece uma DBAL, o que significa que, independentemente de qual banco de dados, o desenvolvedor utiliza as mesmas funções para emitir consultas e buscar dados. O PDO não fornece uma abstração de SQL e nem emula os recursos ausentes de um banco de dados.

## Modelo Relacional

- O modelo relacional é uma abordagem para gerenciar dados usando uma estrutura e linguagem consistente com lógica de predicados de primeira ordem, descrita pela primeira vez em 1969 pelo cientista inglês Edgar F. Codd, onde os dados são representados em termos de tuplas (linhas), agrupadas em relações (tabelas).
- Um banco de dados organizado em termos do modelo relacional é um banco de dados relacional.
- A maioria dos bancos de dados relacionais utiliza o SQL para definição de dados e a linguagem de consulta; esses sistemas implementam o que pode ser considerado como uma aproximação de engenharia ao modelo relacional.
- Além disso, os banco de dados permitem também criar relacionamento entre as tabelas (entidades) através de chaves estrangeiras.

## Modelo Relacional: Tabela, colunas, tipos e chave primária

- Uma tabela é composta de colunas (campos).
- Cada coluna possui um nome e um tipo de dado, ex.: VARCHAR (string), INT, DOUBLE etc.
- É possível definir também se um campo poderá permitir valores nulos ou não.
- Para garantir o unicidade de cada tupla (registro) é possível utilizar o recurso da **chave primária** onde o SGBD garante que aquele valor não possa ser repetido

## Relacionamentos e restrições de integridade

- Um relacionamento, no contexto de bancos de dados relacionais, é uma situação que existe entre duas tabelas quando uma possui uma **chave estrangeira** que faz referência à chave primária da outra tabela. Os relacionamentos permitem que bancos de dados relacionais dividam e armazenem dados em diferentes tabelas, enquanto ligam itens de dados distintos.
- Existem os seguintes tipos de relacionamentos entre tabelas:
  - 1..1 (um para um): onde uma das duas tabelas faz referência para a outra.
  - 1..N (um para muitos): onde o lado N recebe faz referência para o lado um.
  - N..N (muitos para muitos): onde uma terceira tabela (associativa) precisa ser criada para permitir que está faça referência para as duas tabelas da relação.
  - Auto relacionamento: onde uma tabela faz referência para ela mesma.

## Drive PDO

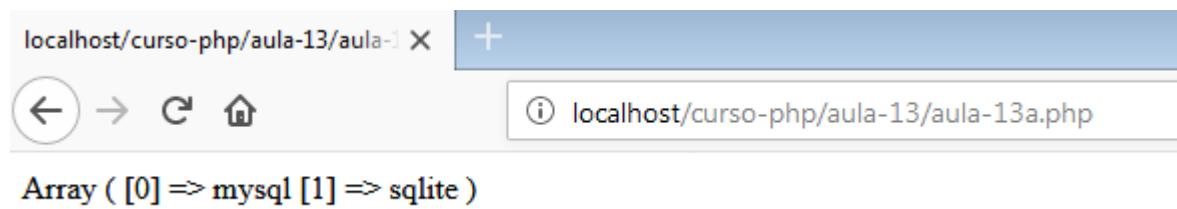
- Um drive/extensão PDO define uma interface leve e consistente para acessar bancos de dados no PHP. Cada driver de banco de dados que implementa a interface do PDO pode expor recursos específicos do banco de dados como funções de extensão regulares.
- Os drivers PDO também são extensões PECL e podem ser habilitadas no php.ini, caso as mesmas estejam disponíveis em forma de \*.dll ou \*.so no diretório de extensões do PHP.

## PDO (getAvailableDrivers())

O método estático getAvailableDrivers permite verificar quais os drivers de PDO estão instalados no PHP.

### aula-13/aula13-a.php

```
<?php  
print_r(PDO::getAvailableDrivers());
```



localhost/curso-php/aula-13/aula13-a.php

Array ( [0] => mysql [1] => sqlite )

## PDO Drivers

Banco (SGBD)	Drive (.dll ou .so)	Incluído no pacote PHP Windows?	Download adicional
MySQL 	php_pdo_mysql	SIM	-
Postgres 	php_pdo_pgsql	SIM	-
SQL Server 	php_pdo_sqlsrv	NÃO	<a href="#">Drive ODBC</a>
Oracle 	php_pdo_oci	SIM	<a href="#">Instant Client</a>

# Instant Client Installation for Microsoft Windows 32-bit

See the [Instant Client Home Page](#) for more information about Instant Client packages.

Client-server version interoperability is detailed in [Doc ID 207303.1](#). For example, Oracle Call Interface 18.3 and 12.2 can connect to Oracle Database 11.2 or later. Some tools may have other restrictions.

1. Download the appropriate Instant Client packages for your platform. All installations require the Basic or Basic Light package.
2. Unzip the packages into a single directory such as `c:\oracle\instantclient_12_2`
3. Add this directory to the `PATH` environment variable. If you have multiple versions of Oracle libraries installed, make sure the new directory occurs first in the path
4. Download and install the correct Visual Studio Redistributable from Microsoft. Instant Client 12.2 requires the [Visual Studio 2013 redistributable](#). Instant Client 12.1 requires the [Visual Studio 2010 redistributable](#). Instant Client 11.2 requires the [Visual Studio 2005 redistributable](#).
5. If you intend to co-locate optional Oracle configuration files such as `tnsnames.ora`, `sqlnet.ora`, `ldap.ora`, or `oraaccess.xml` with Instant Client, then create a subdirectory `c:\oracle\instantclient_12_2\network\admin`

This is the default Oracle client configuration directory for applications linked with this Instant Client.

Alternatively, Oracle client configuration files can be put in another, accessible directory. Then set the environment variable `TNS_ADMIN` to that directory name.



6. Start your application.

ODBC users should follow the [ODBC Installation Instructions](#).

Quando os 4 drivers estiverem corretamente instalados:

```
localhost/curso-php/aula-13/aula-13a.php
Array ( [0] => mysql [1] => oci [2] => pgsql [3] => sqlsrv [4] => sqlite )
```

## Classe PDO (instanciando)

```
<?php  
$pdo = new PDO($dsn, $usuario, $senha, $opcoes);
```

- **dsn**: conexão com a fonte de banco de dados
- **usuario**: usuário do banco de dados
- **senha**: senha deste usuário
- **opções**: conjunto de opções em forma de chave e valores utilizando um conjunto de constantes, alguns comuns entre os drivers e outros específicos de cada um.

## PDO Options (opções recomendadas)

Opção	Valor Recomendado	Resultado	Suportado pelos Drivers
PDO::ATTR_ERRMODE	PDO::ERRMODE_EXCEPTION	Lança Exceptions toda vez que uma instrução SQL falhar.	MySQL, Postgres, Sql Server e Oracle
PDO::ATTR_EMULATE_PREPARES	false	O motor do banco de dados fará o <i>prepared statement</i> ao em vez do PDO e assim, consulta e os dados reais são enviados separadamente, aumentando a segurança.	MySQL, Postgres e Oracle
PDO::ATTR_DEFAULT_FETCH_MODE	PDO::FETCH_ASSOC	É conveniente configurá-lo de forma global e depois omiti-lo em buscas específicas.	MySQL, Postgres, Sql Server e Oracle

## DSN (Data Source Name)

- DSN ou data source name (nome de fonte de dados, algumas vezes conhecido como nome de fonte de banco de dados, apesar de fontes de dados não serem limitadas a bancos de dados), é uma estrutura de dados usada para descrever uma conexão a uma fonte de dados.
- Cada Driver possui seu conjunto específico de parâmetros. Exemplos:
  - **MYSQL:** mysql:host=localhost;dbname=livraria;port=3306;charset=utf8mb4
  - **POSTGRES:** pgsql:host=localhost;port=5432;dbname=livraria;
  - **SQL SERVER:** sqlsrv:Server=localhost;Database=livraria
  - **ORACLE:**  
oci:dbname=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))(CONNECT \_DATA=(SERVICE\_NAME=LIVRARIA)))

■ Nome ou IP da máquina onde o Banco (SGBD) está executando.

■ Nome da base onde as tabelas se encontram.

■ Porta de conexão do SGBD

## PDO (Conexão MySQL)

### aula-13/pdo/conexao.php

```
<?php

$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES => false, //para funcionar bind no limit
];

$servidor = "localhost";
$banco = "livraria";
$usuario = "root";
$senha = "";
$porta = 3306;

$dsn = "mysql:host=$servidor;port=$porta;dbname=$banco;charset=utf8";

$pdo = new PDO($dsn, $usuario, $senha, $options);

var_dump($pdo);
```



object(PDO)#1 (0) { }

## PDO (Conexão Postgres)

### aula-13/pdo/conexao\_postgres.php

```
<?php

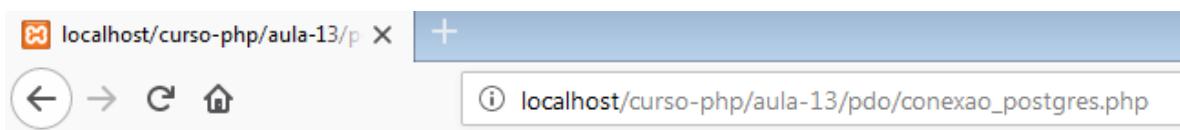
$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES => false,
];

$servidor = "localhost";
$banco = "livraria";
$usuario = "postgres";
$senha = "admin";
$porta = 5432;

$dsn = "pgsql:host=$servidor;port=$porta;dbname=$banco;";

$pdo = new PDO($dsn, $usuario, $senha, $options);

var_dump($pdo);
```



object(PDO)#1 (0) { }

## PDO (Conexão SQL Server)

### **aula-13/pdo/conexao\_sqlserver.php**

```
<?php

$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION, //ver erros de query
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
];

$servidor = "localhost";
$banco = "livraria";
$usuario = "85bits";
$senha = "admin";

$dsn = "sqlsrv:Server=$server;Database=$banco";

$pdo = new PDO($dsn, $usuario, $senha, $options);
```

## PDO (Conexão Oracle)

### aula-13/pdo/conexao\_oracle.php

```
<?php

$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_EMULATE_PREPARES => false,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_CASE => PDO::CASE_LOWER
];

$servidor = "localhost";
$usuario = "php";
$senha = "admin";
$service_name = "XE";
$sid = "XE";
$port = 1521;

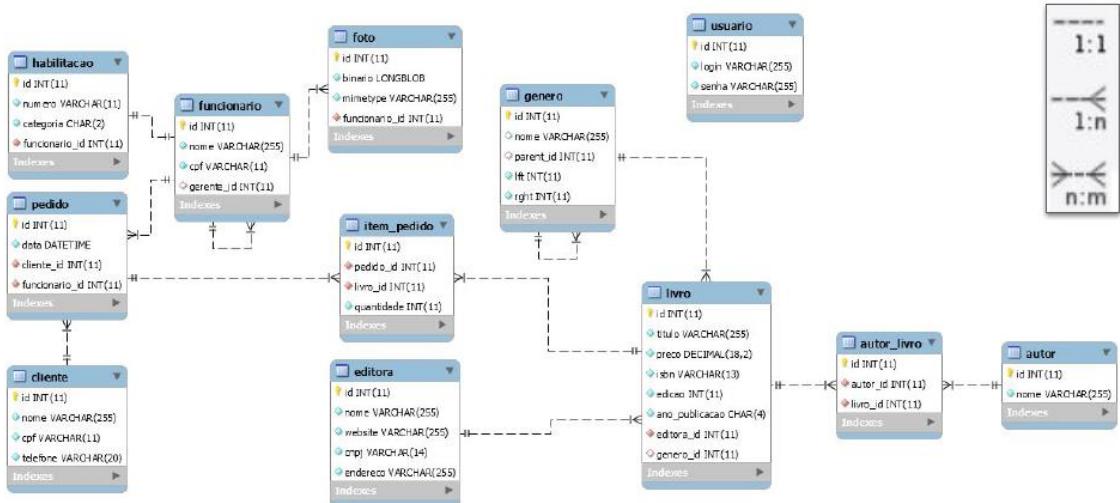
$dbtns = "(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST =
$servidor)(PORT = $port)) (CONNECT_DATA =
(SERVICE_NAME = $service_name) (SID = $sid)))";

$pdo = new PDO("oci:dbname=" . $dbtns . ";charset=utf8", $usuario, $senha,
$options);
```

## Modelo de Banco de Dados Livraria

- Para os exemplos deste curso utilizaremos um banco de dados do domínio de uma Livraria. O banco está disponível para 4 SGBDS diferentes e já conta com um número considerável de registros nas tabelas.
- Além disso, este banco possui todos os tipos de associações:
  - a. (1..1): Funcionário tem uma Habilidade.
  - b. (1..N): Editora tem muitos Livros.
  - c. (N..1): Livros pertencem a uma Editora.
  - d. (N..N): Livros têm muitos Autores e Autores têm muitos Livros.
- E algumas Associações especiais:
  - a. Auto relacionamento: Funcionário tem gerente Gerente (Funcionário).
  - b. N..N com dados associativos: Pedido tem muitos Livros (através de ItemPedido e seus dados).
  - c. Árvore: Gênero tem nós filhos, irmãos e pais.

# Modelo de Exemplo (livraria)



## SQL (livraria v0.2 - DUMP do Banco de Dados)

Disponível em:



<https://gist.github.com/celsowm/5a3723b58775900db7d010653e5f82e0>



<https://gist.github.com/celsowm/067fe51dfa612697895c8ec3b5cb436d>



<https://gist.github.com/celsowm/b139713d65d6c42df084269b3f150a2d>



<https://gist.github.com/celsowm/219c130a18289b9378fa7642508c473b>

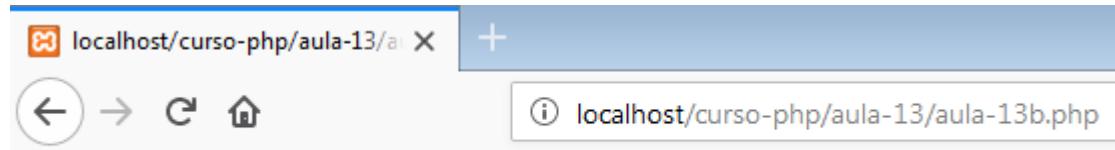
## PDO Query (PDO->query() e PDOStatement->fetch())

O método query permite submeter uma query SQL para o SGBD e retorna um objeto do tipo PDOStatement.

Para recuperar (a próxima/primeira) linha do resultado da query podemos utilizar o método fetch(), que, no caso do fetch padrão associativo, retorna um array onde as chaves são as colunas e os valores de cada linha os valores do array.

### **aula-13/aula-13b.php**

```
<?php  
  
include_once "pdo/conexao.php";  
  
$statement = $pdo->query("SELECT nome FROM funcionario");  
$funcionario = $statement->fetch();  
  
echo $funcionario['nome'];
```

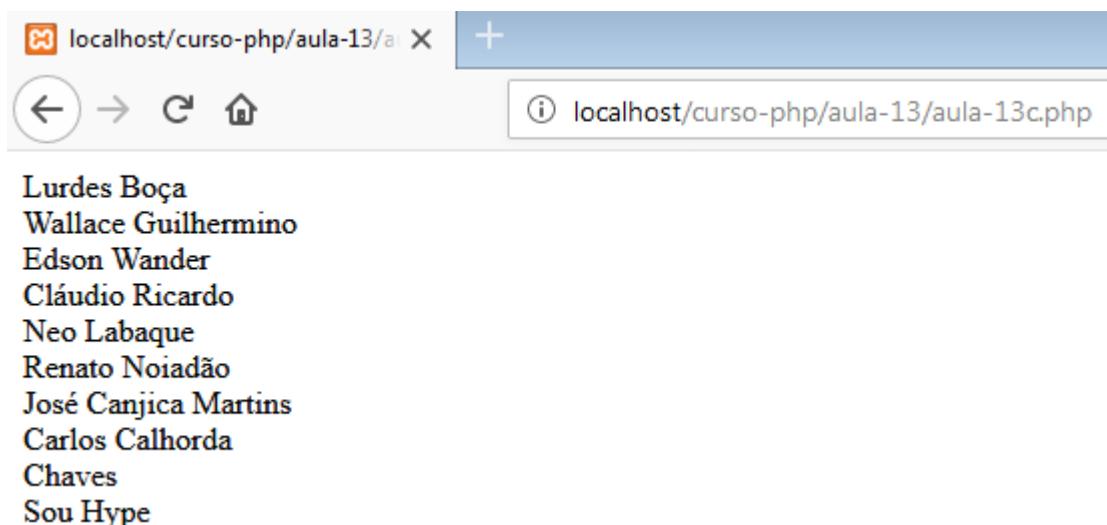


## PDO Query (PDO->query() e PDOStatement->fetch())

Podemos iterar o resultado de uma query invocando o método fetch até o mesmo retornar nulo, isto é, até esgotar o número de registros.

### aula-13/aula-13c.php

```
<?php  
  
include_once "pdo/conexao.php";  
  
$statement = $pdo->query("SELECT nome FROM funcionario");  
  
while($funcionario = $statement->fetch()){  
    echo $funcionario['nome']."<br>";  
}  
}
```

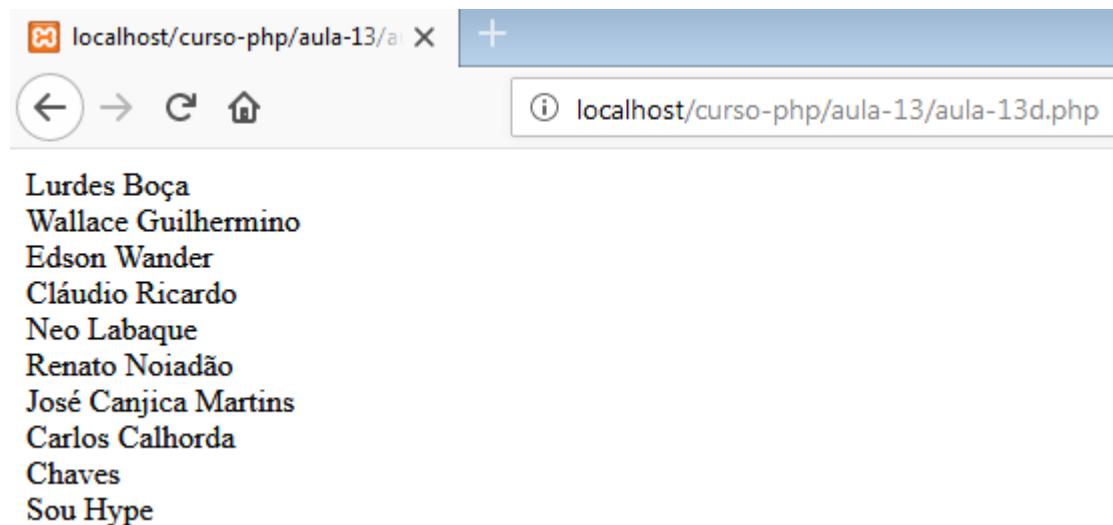


## PDO Query (Transversible)

A classe PDOStatement implementa Traversable, o que permite que objetos desta classe possam ser iterados de forma transparente.

### aula-13/aula-13d.php

```
<?php  
  
include_once "pdo/conexao.php";  
  
$statement = $pdo->query('SELECT nome FROM funcionario');  
  
foreach ($statement as $linha){  
    echo $linha['nome'] . "<br>";  
}
```



# Aula 14 - SQL Injection, Prepared Statement, FetchAll, like, in

## SQL Injection

"SQL Injection" é um subconjunto da vulnerabilidade de entrada do usuário não verificada / não-tratada cujo propósito é forçar o aplicativo a executar um código SQL que não foi planejado.

Exemplo:

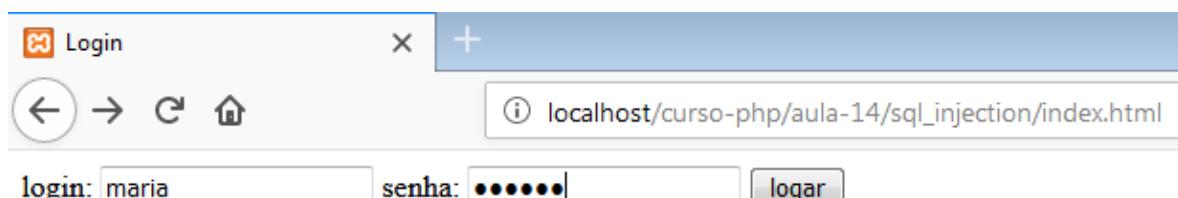
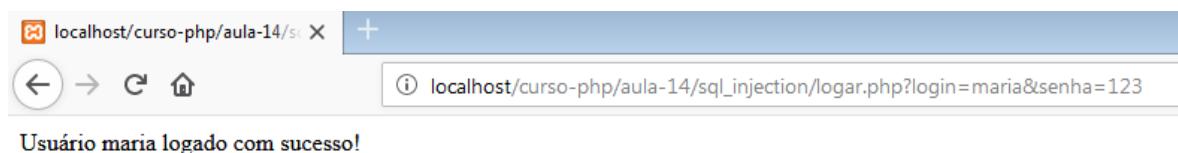
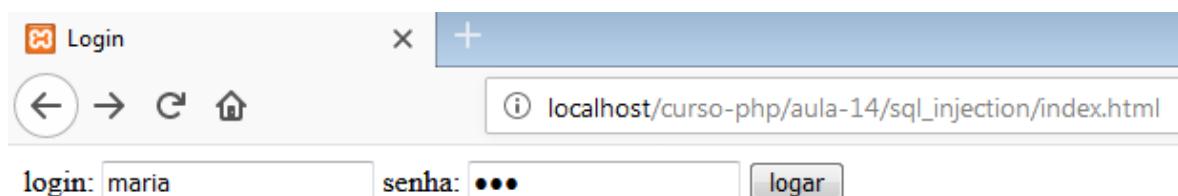
' or '1'='1

### **aula-14/sql\_injection/index.html**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Login</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <form action="logar.php">
      <label>login:</label>
      <input type="text" name="login" />
      <label>senha:</label>
      <input type="password" name="senha" />
      <input type="submit" value="logar" />
    </form>
  </body>
</html>
```

## aula-14/sql\_injection/logar.php

```
<?php  
  
include_once '../conexao.php';  
  
$login = $_REQUEST['login'];  
$senha = $_REQUEST['senha'];  
  
$query = "SELECT * FROM usuario WHERE login = '$login' AND senha = '$senha"';  
  
//var_dump($query);  
  
$statement = $pdo->query($query);  
$usuario = $statement->fetch();  
  
if($usuario){  
    echo "Usuário {$usuario['login']} logado com sucesso!";  
} else {  
    echo "Usuário não encontrado ou senha errada!";  
}
```



## Injeção de SQL

The screenshot shows a login page with two input fields: 'login' containing "' or '1='1" and 'senha' containing a series of dots. A 'logar' button is present. Below the form, a success message 'Usuário maria logado com sucesso!' is displayed. The URL in the address bar is localhost/curso-php/aula-14/sql\_injection/logar.php?login=' or '1'%3D'1&senha=' or '1'%3D'1'. To the right of the message, a SQL query is shown in a code block:

```
SELECT * FROM usuario  
WHERE login = '1 OR '1' = '1' AND senha = '' OR '1' = '1'
```

## Prepared Statements

- Para evitar SQL Injection podemos utilizar o recurso de prepared statement (declaração/instrução preparada/parametrizada).
- Os prepared statements são resilientes à SQL injections porque os valores que são transmitidos posteriormente usando um protocolo diferente e não são compilados/interpretados com o código SQL original.
- Para utilizar este recurso com o PDO devemos utilizar o método `prepare()` com a query desejada substituindo os valores por placeholders (caracteres substitutos).
- Antes de recuperar os dados (`fetch`) faz-se necessário vincular os valores aos seus respectivos placeholders e executar (`execute`).
- Os Drivers PDO podem utilizar prepared statements de forma nativa (quando suportado) ou emulados pelo PDO (`PDO::ATTR_EMULATE_PREPARES`)

## Evitando SQL Injection

### aula-14/sql\_injection/index1.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Login</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <form action="logar_prepared.php">
      <label>login:</label>
      <input type="text" name="login" />
      <label>senha:</label>
      <input type="password" name="senha" />
      <input type="submit" value="logar" />
    </form>
  </body>
</html>
```

### aula-14/sql\_injection/logar\_prepared.php

```
<?php

include_once '../conexao.php';

$login = $_REQUEST['login'];
$senha = $_REQUEST['senha'];

$query = "SELECT * FROM usuario WHERE login = ? AND senha = ?";

$statement = $pdo->prepare($query);
$statement->execute([$login, $senha]);
$usuario = $statement->fetch();

if($usuario){
  echo "Usuário {$usuario['login']} logado com sucesso!";
} else {
  echo "Usuário não encontrado ou senha errada!";
}
```

localhost/curso-php/aula-14/sql\_injection/index1.html

login: ' or '1'='1

senha: XXXXXXXXXX

logar

localhost/curso-php/aula-14/sql\_injection/logar\_prepared.php?login='+or+'1%3D'1&senha='+or+'1%3D'1

Usuário não encontrado ou senha errada!

## PDO Binding (usando parâmetros no SQL)

### aula-14/aula-14a.php

```
<?php
include_once "conexao.php";

$nome = 'Edson Wander';
$cpf = '54698715324';

//posicional

$statement = $pdo->prepare('SELECT * FROM funcionario WHERE nome = ? AND
cpf = ?');
$statement->execute([$nome, $cpf]);
$funcionario = $statement->fetch();

if($funcionario){
    echo "Usuário {$funcionario['nome']} logado com sucesso!";
} else {
    echo "Usuário não encontrado ou senha errada!";
}

?>
```

localhost/curso-php/aula-14/aula-14a.php

Usuário Edson Wander logado com sucesso!

## Parâmetros nomeados

### aula-14/aula-14b.php

```
<?php

include_once "conexao.php";

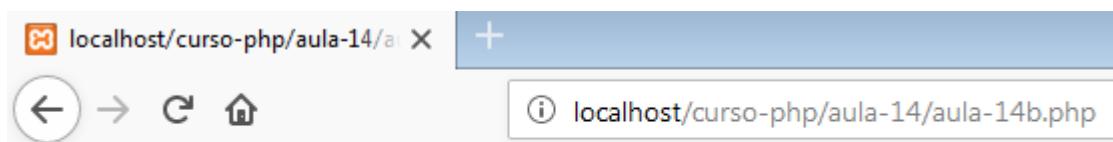
$nome = 'Edson Wander';
$cpf = '54698715324';

//usando key e value

$stmt = $pdo->prepare('SELECT * FROM funcionario WHERE nome = :nome
AND cpf = :cpf');
$stmt->execute(['nome' => $nome, 'cpf' => $cpf]);
$funcionario = $stmt->fetch();

if($funcionario){
    echo "Usuário {$funcionario['nome']} logado com sucesso!";
} else {
    echo "Usuário não encontrado ou senha errada!";
}

?>
```



Usuário Edson Wander logado com sucesso!

## PDO Binding (bindParam)

Podemos utilizar o método bindParam para passar por referência o valor de variáveis para uma prepared statement.

No primeiro parâmetro especificamos o nome ou posição do placeholder; no segundo, a variável de referência e no terceiro o tipo desejado do valor (usando constantes do PDO).

### aula-14/aula-14c.php

```
<?php

include 'conexao.php';

$nome = 'Edson Wander';
$cpf = '54698715324';

$stmt = $pdo->prepare('SELECT * FROM funcionario WHERE nome = ? AND cpf = ?');

$stmt->bindParam(1, $nome,PDO::PARAM_STR);
$stmt->bindParam(2, $cpf,PDO::PARAM_STR);
$stmt->execute();

$funcionario = $stmt->fetch();

echo "<pre>";
print_r($funcionario);
echo "</pre>";
```

```
localhost/curso-php/aula-14/aula-14c.php

Array
(
    [id] => 9
    [nome] => Edson Wander
    [cpf] => 54698715324
    [gerente_id] => 8
)
```

## PDO Binding (bindValue)

Podemos utilizar o método bindValue para passar por um valor de variáveis para uma prepared statement.

No primeiro parâmetro especificamos o nome ou posição do placeholder; no segundo, a variável de referência e no terceiro o tipo desejado do valor (usando constantes do PDO).

### aula-14/aula-14d.php

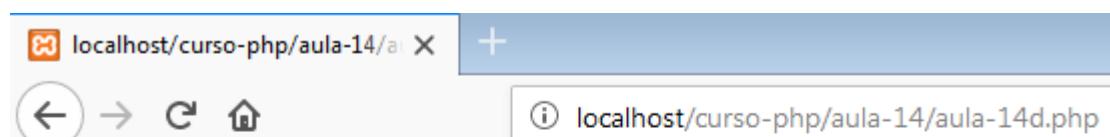
```
<?php
include 'conexao.php';

$stmt2 = $pdo->prepare('SELECT * FROM funcionario WHERE nome = ? AND cpf = ?');

$stmt2->bindValue(1, 'Edson Wander', PDO::PARAM_STR);
$stmt2->bindValue(2, '54698715324', PDO::PARAM_STR);
$stmt2->execute();

$funcionario2 = $stmt2->fetch();

echo "<pre>";
print_r($funcionario2);
echo "</pre>";
```



```
Array
(
    [id] => 9
    [nome] => Edson Wander
    [cpf] => 54698715324
    [gerente_id] => 8
)
```

## PDO (bindColumn)

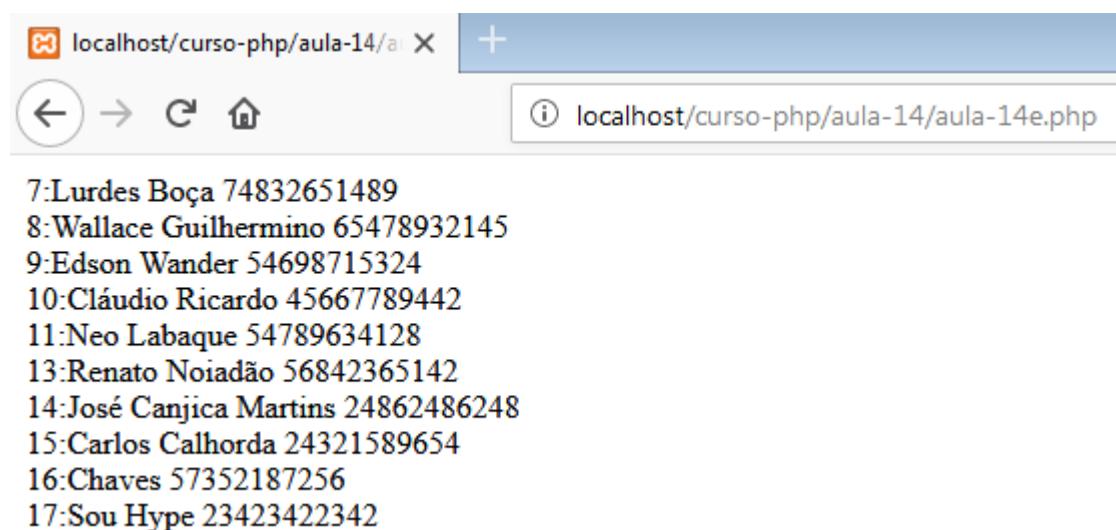
Com o uso do bindColumn é possível passar por referência o valor de uma coluna para uma variável para cada fetch realizado.

A indicação da coluna pode ser feita de maneira posicional ou pelo nome da coluna.

O ideal é utilizar sempre o estilo PDO::FETCH\_BOUND que permite que o PDO possa designar valores para variáveis que foram “vinculadas” anteriormente usando bindColumn.

### aula-14/aula-14e.php

```
<?php  
include 'conexao.php';  
  
$query = "SELECT id, nome, cpf FROM funcionario";  
  
$statement = $pdo->query($query);  
$statement->bindParam(1, $id);  
$statement->bindParam(2, $nome);  
$statement->bindParam('cpf', $cpf);  
  
while ($row = $statement->fetch(PDO::FETCH_BOUND)) {  
    echo "$id:" . $nome . " " . $cpf . "<br>";  
}  
?
```



7:Lurdes Boça	74832651489
8:Wallace Guilhermino	65478932145
9:Edson Wander	54698715324
10:Cláudio Ricardo	45667789442
11:Neo Labaque	54789634128
13:Renato Noiadão	56842365142
14:José Canjica Martins	24862486248
15:Carlos Calhorda	24321589654
16:Chaves	57352187256
17:Sou Hype	23423422342

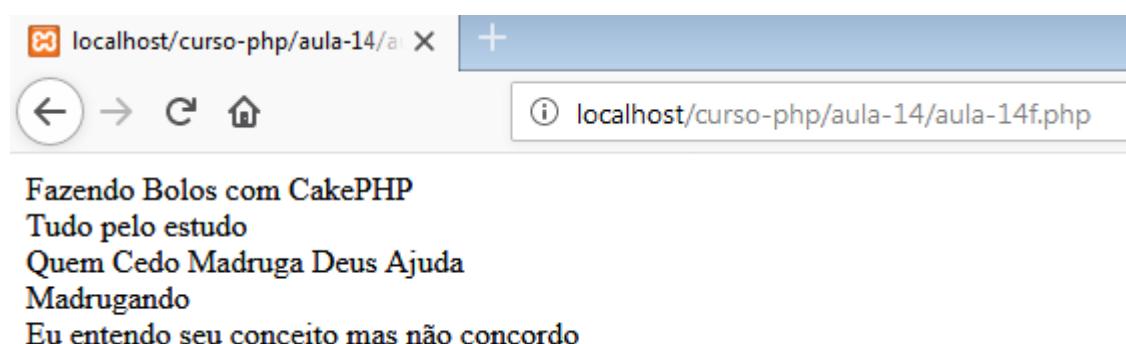
## PDO Prepared Statement (com SQL like)

O operador like do SQL, que permite procurar por um determinado “padrão” em um texto, pode ser utilizado também como prepared statement.

Só é válido salientar que os operadores curingas precisam ser utilizados “fora” da query, isto é, precisam ser enviados como valores para os binds.

### aula-14/aula-14f.php

```
<?php  
  
include 'conexao.php';  
  
$sql = "SELECT * FROM livro WHERE titulo LIKE ?";  
  
$statement = $pdo->prepare($sql);  
$statement->execute(['%do%']);  
  
foreach($statement as $livro){  
    echo $livro['titulo']."<br>";  
}
```



## PDO Prepared Statement (com SQL IN() "literal")

Infelizmente o uso de array como placeholders não é suportado nativamente no PDO, então, faz-se necessário “replicar” um conjunto de placeholders que possa representar cada valor do conjunto do IN().

É válido salientar que muitos SGBDs possuem limitação no número de valores literais em um IN.

### aula-14/aula-14g.php

```
<?php
include 'conexao.php';

$filtro = ["preco_minimo" => "1.98"];
$edicoes = [1,2,10];
$edicoes = array_combine(
    array_map(function($i){ return ':id'.$i; }, array_keys($edicoes)), $edicoes
);

$in_placeholders = implode(',', array_keys($edicoes));

$sql = "SELECT * FROM livro WHERE preco >= :preco_minimo AND edicao IN
($in_placeholders)";

$stmt = $pdo->prepare($sql);
$stmt->execute(array_merge($filtro,$edicoes));

foreach($stmt as $livro){
    echo "Título: " . $livro['titulo'] . " - Edição: " . $livro['edicao'] . " - Preço: " .
    $livro['preco'] . "</br>";
}
```



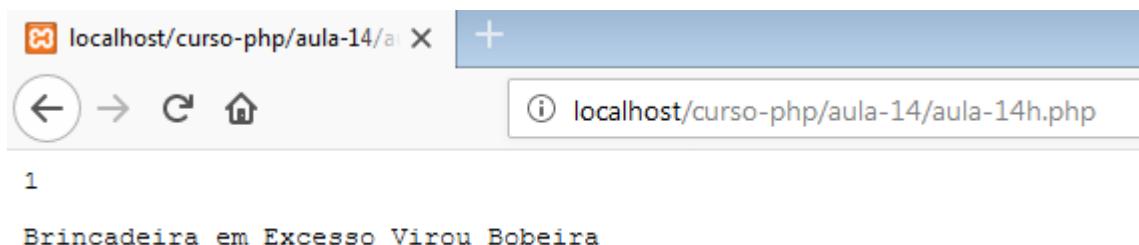
Título: Brincadeira em Excesso Virou Bobeira - Edição: 2 - Preço: 44.01  
Título: Vamos Investigar? - Edição: 2 - Preço: 63.22  
Título: Portabilidade Manual: Um Tutorial Prático - Edição: 2 - Preço: 100.99  
Título: Brazil Mulambo - Edição: 1 - Preço: 9.99  
Título: Tudo pelo estudo - Edição: 1 - Preço: 1.99  
Título: Madrugando - Edição: 10 - Preço: 18.89

## PDO (fetchColumn)

O método fetchColumn retorna uma única coluna da próxima linha de um conjunto de resultados ou FALSE se não houver mais linhas.

### aula-14/aula-14h.php

```
<?php  
  
include_once "conexao.php";  
  
//PDO fetchColumn  
  
$statement = $pdo->query("SELECT id, titulo FROM livro");  
  
echo "<pre>";  
print_r($statement->fetchColumn());  
echo "</pre>";  
  
echo "<pre>";  
print_r($statement->fetchColumn(1));  
echo "</pre>";
```



## PDO getColumnData (Introspecção)

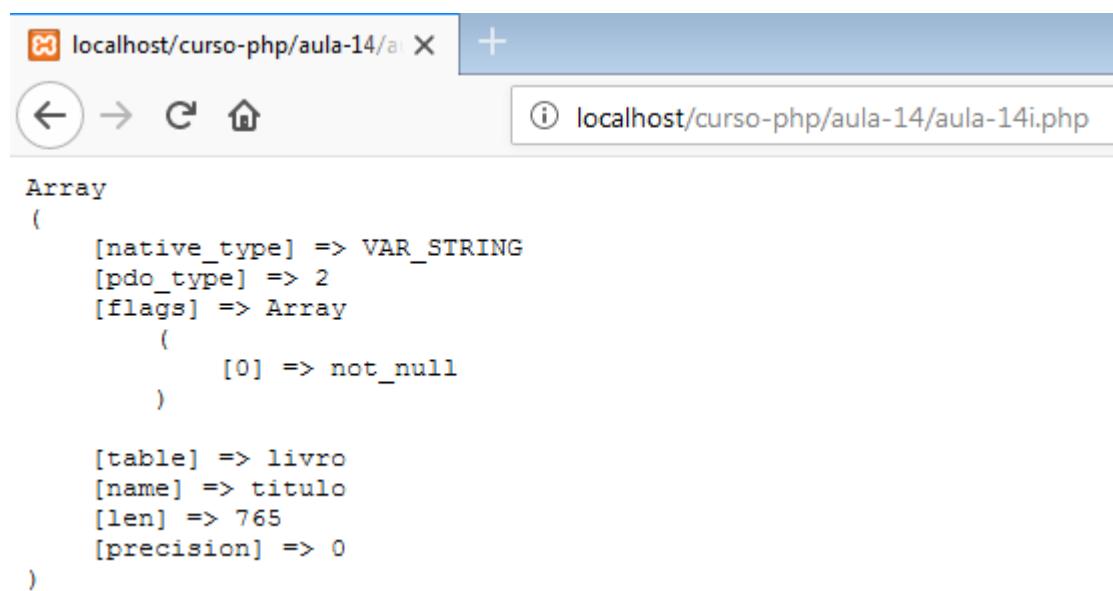
O método getColumnData permite que recuperar os metadados dos campos (colunas) de uma query (inclusive campos virtuais).

O parâmetro do método é a posição da coluna em relação do descrito na query.

Infelizmente o drive do Oracle (pdo\_oci) não suporta/implementa este método.

### aula-14/aula-14i.php

```
<?php  
  
include 'conexao.php';  
  
$statement = $pdo->query('SELECT titulo, preco FROM livro');  
$metadados = $statement->getColumnMeta(0);  
  
echo "<pre>";  
print_r($metadados);  
echo "</pre>";
```



```
Array
(
    [native_type] => VAR_STRING
    [pdo_type] => 2
    [flags] => Array
        (
            [0] => not_null
        )

    [table] => livro
    [name] => titulo
    [len] => 765
    [precision] => 0
)
```

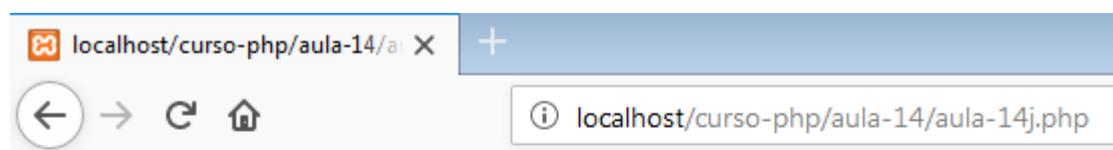
## PDO FetchAll

- PDOStatement::fetchAll() retorna um array contendo todas as linhas restantes no conjunto de resultados. O array representa cada linha como uma matriz de valores de coluna ou um objeto com propriedades correspondentes a cada nome de coluna.
- Uma array vazio é retornado se houver zero resultados a serem obtidos ou retorna FALSE em caso de falha.
- Usar esse método para buscar conjuntos de resultados grandes resultará em uma grande demanda no sistema e possivelmente nos recursos da rede. Em vez de recuperar todos os dados e manipulá-los no PHP, considere o uso do servidor de banco de dados para manipular os conjuntos de resultados. Por exemplo, use as cláusulas WHERE ou Paginação no SQL para restringir os resultados antes de recuperá-los e processá-los com o PHP.

O resultado do fetchAll() é um array com o resultado da query.

### aula-14/aula-14j.php

```
<?php  
  
include_once "conexao.php";  
  
$statement = $pdo->query('SELECT * FROM funcionario');  
$funcionarios = $statement->fetchAll();  
  
foreach ($funcionarios as $funcionario) {  
    echo $funcionario['nome']."<br>";  
}
```



Lurdes Boça  
Wallace Guilhermino  
Edson Wander  
Cláudio Ricardo  
Neo Labaque  
Renato Noiadão  
José Canjica Martins  
Carlos Calhorda  
Chaves  
Sou Hype

# Aula 15 - PDO Todos os Modos de Recuperação de Dados + Benchmark Lazy + ScrollCursor

## Fetch & Fetch All (estilos)

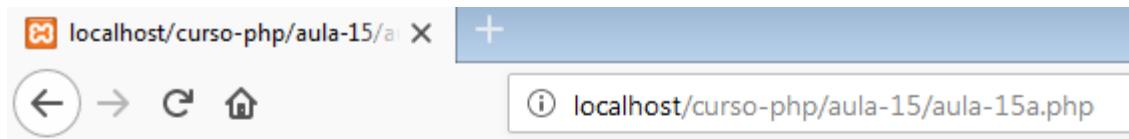
- Ao utilizar os métodos fetch ou fetchAll de um PDOStatement é possível determinar qual o “formato” dos dados resultantes da query.
- Alguns estilos possuem argumentos (similar a uma função). E os valores destes argumentos influenciam na formatação dos dados.
- O estilo dos fetchs podem ser determinados das seguintes maneiras :
  - De maneira global no options do PDO (PDO::ATTR\_DEFAULT\_FETCH\_MODE)
  - Através do segundo parâmetro dos métodos fetch e fetchAll do PDOStatement
  - Utilizando o método setFetchMode() em cada PDOStatement antes do fetch/fetch/execute.

## PDO::FETCH\_ASSOC

PDO::FETCH\_ASSOC: retorna um array indexado pelo nome da coluna conforme o retorno da query.

### aula-15/aula-15a.php

```
<?php  
include 'conexao.php';  
  
$statement = $pdo->query('SELECT nome FROM funcionario');  
$funcionarios = $statement->fetchAll(PDO::FETCH_ASSOC);  
  
foreach ($funcionarios as $funcionario) {  
    echo $funcionario['nome']."<br>";  
}  
  
$statement = $pdo->query('SELECT nome FROM funcionario');  
$funcionario = $statement->fetch(PDO::FETCH_ASSOC);  
  
echo "<br>" . $funcionario['nome']."<br>";
```



Lurdes Boça  
Wallace Guilhermino  
Edson Wander  
Cláudio Ricardo  
Neo Labaque  
Renato Noiadão  
José Canjica Martins  
Carlos Calhorda  
Chaves  
Sou Hype

Lurdes Boça

## PDO::FETCH\_NUM

PDO::FETCH\_NUM: retorna um array 2D onde as chaves assumem as posições das colunas e os valores são os dados de cada registro nesta coluna.

### aula-15/aula-15b.php

```
<?php
include 'conexao.php';

$statement = $pdo->query("SELECT nome, id FROM funcionario");

// $statement = $pdo->query("SELECT titulo, id FROM livro");
$dados = $statement->fetchall(PDO::FETCH_NUM);

foreach ($dados as $dado) {
    echo "nome: $dado[0] | id: $dado[1] <br/>";
}
```

```
nome: Lurdes Boça | id: 7
nome: Wallace Guilhermino | id: 8
nome: Edson Wander | id: 9
nome: Cláudio Ricardo | id: 10
nome: Neo Labaque | id: 11
nome: Renato Noiadão | id: 13
nome: José Canjica Martins | id: 14
nome: Carlos Calhorda | id: 15
nome: Chaves | id: 16
nome: Sou Hype | id: 17
```

## PDO::FETCH\_BOTH

PDO::FETCH\_BOTH: retorna um array indexado duplicando o número colunas onde permitindo que os dados sejam acessados tanto pelo nome da coluna como pelo índice (número) da mesma.

### aula-15/aula-15c.php

```
<?php
include 'conexao.php';

$statement = $pdo->query('SELECT * FROM funcionario');
$funcionario = $statement->fetch(PDO::FETCH_BOTH);

echo "<pre>";
print_r($funcionario);
echo "</pre>";
```

```
Array
(
    [id] => 7
    [0] => 7
    [nome] => Lurdes Boça
    [1] => Lurdes Boça
    [cpf] => 74832651489
    [2] => 74832651489
    [gerente_id] =>
    [3] =>
)
```

## PDO::FETCH\_KEY\_PAIR

PDO::FETCH\_KEY\_PAIR: retorna um array 2D onde as chaves assumem os valores da primeira coluna e os valores são os dados da segunda.

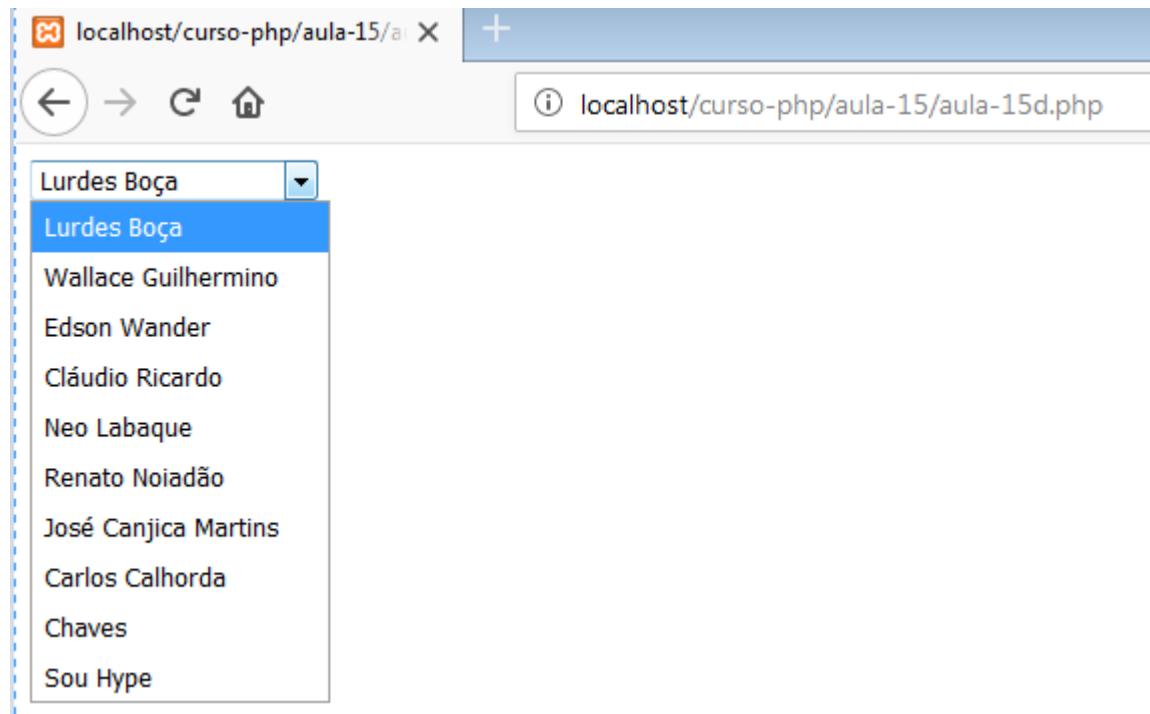
Se uma terceira coluna for colocada na projeção, o PDO lançará uma exceção.

### aula-15/aula-15d.php

```
<?php
include_once "conexao.php";

$statement = $pdo->query("SELECT id, nome FROM funcionario");
$dados = $statement->fetchall(PDO::FETCH_KEY_PAIR);

echo "<select>";
foreach($dados as $key => $dado){
    echo "<option value='".$key."'>$dado</option>";
}
echo "</select>";
```



The screenshot shows a web browser window with the URL `localhost/curso-php/aula-15/aula-15d.php`. A dropdown menu is open, displaying a list of names. The first item, "Lurdes Boça", is highlighted with a blue selection bar. The other items in the list are: Wallace Guilhermino, Edson Wander, Cláudio Ricardo, Neo Labaque, Renato Noiadão, José Canjica Martins, Carlos Calhorda, Chaves, and Sou Hype.

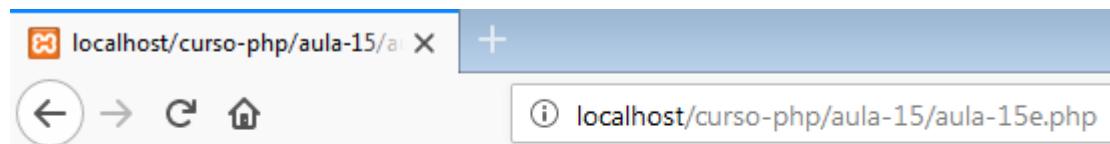
Nome
Lurdes Boça
Wallace Guilhermino
Edson Wander
Cláudio Ricardo
Neo Labaque
Renato Noiadão
José Canjica Martins
Carlos Calhorda
Chaves
Sou Hype

## PDO::FETCH\_UNIQUE

PDO::FETCH\_UNIQUE: retorna um array de arrays onde as chaves assumem os valores da primeira coluna e os valores (com colunas em keys) em subarrays.

### aula-15/aula-15e.php

```
<?php  
  
include_once "conexao.php";  
  
$statement = $pdo->query('SELECT id,nome,cpf FROM funcionario');  
$funcionarios = $statement->fetchAll(PDO::FETCH_UNIQUE);  
  
foreach ($funcionarios as $id => $funcionario) {  
    echo $id.":". $funcionario['nome']."</br>";  
}
```



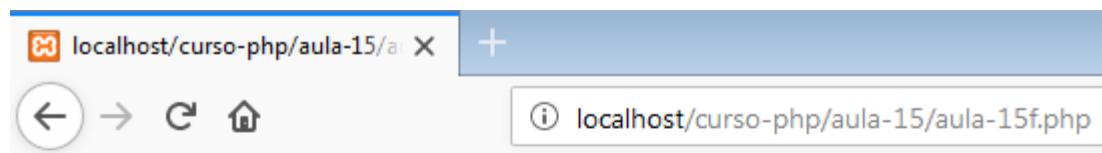
```
7:Lurdes Boça  
8:Wallace Guilhermino  
9:Edson Wander  
10:Cláudio Ricardo  
11:Neo Labaque  
13:Renato Noiadão  
14:José Canjica Martins  
15:Carlos Calhorda  
16:Chaves  
17:Sou Hype
```

## PDO::FETCH\_NAMED

PDO::FETCH\_NAMED: retorna um array multidimensional onde campos com nomes repetidos são colocados dentro de um mesmo array onde, a ordem dos mesmos é referente a ordem de suas respectivas tabelas na própria query.

### aula-15/aula-15f.php

```
<?php  
  
include_once "conexao.php";  
  
$statement = $pdo->query('SELECT * FROM editora LEFT JOIN livro ON  
livro.editora_id = editora.id');  
$registros = $statement->fetchAll(PDO::FETCH_NAMED);  
  
foreach ($registros as $registro) {  
    echo "editora id {$registro['id'][0]} e livro {$registro['id'][1]} <br/>";  
}
```



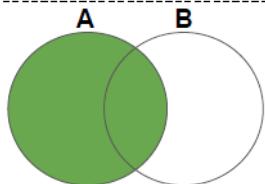
The screenshot shows a web browser window with the URL `localhost/curso-php/aula-15/aula-15f.php` in the address bar. The page content displays a list of strings, each consisting of 'editora id' followed by a number and 'e livro' followed by another number, separated by a space. The numbers correspond to the IDs shown in the previous screenshot's table.

editora id	livro
1	1
1	3
1	6
1	8
1	10
2	2
2	4
2	5
3	7
4	9
5	
6	
7	

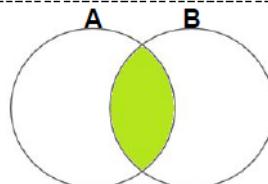
```
editora id 1 e livro 1  
editora id 1 e livro 3  
editora id 1 e livro 6  
editora id 1 e livro 8  
editora id 1 e livro 10  
editora id 2 e livro 2  
editora id 2 e livro 4  
editora id 2 e livro 5  
editora id 3 e livro 7  
editora id 4 e livro 9  
editora id 5 e livro  
editora id 6 e livro  
editora id 7 e livro
```

# JOIN SQL

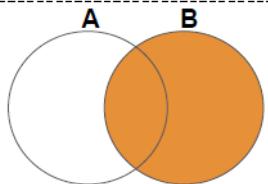
**LEFT JOIN:** junção de A e B, porém, recupere tudo de A mesmo se não houver uma referência em B (ficando nulos valores para B sem referência para A).



**INNER JOIN:** junção de A e B, porém recupere apenas se houver registro A e B com referências.



**RIGHT JOIN:** junção de A e B, porém, recupere tudo de B mesmo se não houver uma referência em A (ficando nulos valores para A sem referência para B).



```
SELECT * FROM funcionario  
LEFT JOIN habilitacao ON  
funcionario.id =  
habilitacao.funcionario_id
```

```
SELECT * FROM pedido  
INNER JOIN cliente ON  
cliente.id = pedido.cliente_id
```

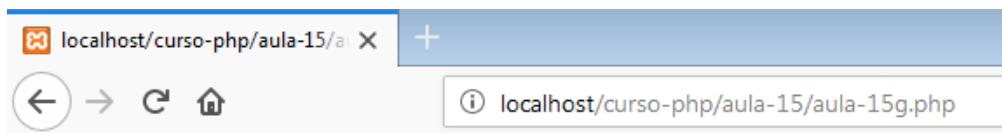
```
SELECT * FROM livro  
RIGHT JOIN editora ON editora.id  
= livro.editora_id
```

## PDO::FETCH\_GROUP

PDO::FETCH\_GROUP: retorna os registros agrupados pela primeira coluna da query em arrays multidimensionais.

### aula-15/aula-15g.php

```
<?php  
  
include 'conexao.php';  
  
$statement = $pdo->query("SELECT edicao, id, titulo FROM livro order by edicao");  
$livrosPorEdicao = $statement->fetchAll(PDO::FETCH_GROUP);  
  
foreach ($livrosPorEdicao as $key => $livros) {  
    echo "Livros na $key ª edição: <br>";  
    foreach ($livros as $livro) {  
        echo "-- {$livro['titulo']} <br>";  
    }  
    echo "<br/>";  
}
```



Livros na 1<sup>a</sup> edição:

- Eu entendo seu conceito mas não concordo
- Tudo pelo estudo
- Brazil Mulambo

Livros na 2<sup>a</sup> edição:

- Vamos Investigar?
- Portabilidade Manual: Um Tutorial Prático
- Brincadeira em Excesso Virou Bobeira

Livros na 3<sup>a</sup> edição:

- Fazendo Bolos com CakePHP

Livros na 5<sup>a</sup> edição:

- Quem Cedo Madruga Deus Ajuda

Livros na 10<sup>a</sup> edição:

- Madrugando

Livros na 15<sup>a</sup> edição:

- Sucesso na Vida

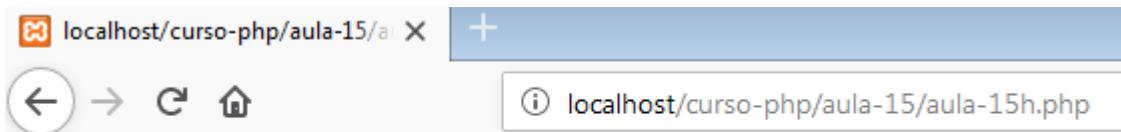
## PDO::FETCH\_LAZY

PDO::FETCH\_LAZY: Faz um busca tardia/lenta/por demanda e retornar a próxima linha como um objeto anônimo com nomes de colunas como atributos. O desempenho costuma ser notável ao recuperar uma grande massa de dados (linhas).

Atenção: Devido à própria natureza de busca “tardia” (e por demanda) dos registros do banco de dados, o FETCH\_LAZY não funciona com o fetchAll().

## aula-15/aula-15h.php

```
<?php  
  
include_once "conexao.php";  
  
$statement = $pdo->query('SELECT * FROM livro');  
$livro = $statement->fetch(PDO::FETCH_LAZY);  
  
echo "ISBN: {$livro['isbn']} : {$livro->título} </br>";
```



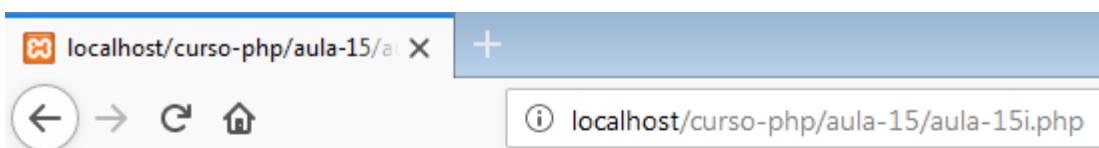
ISBN: 123456789112 : Sucesso na Vida

## PDO::FETCH\_BOUND

PDO::FETCH\_BOUND: Permite que o PDO possa designar valores para variáveis que foram “vinculadas” anteriormente usando bindColumn.

### aula-15/aula-15i.php

```
<?php  
  
include 'conexao.php';  
  
$query = "SELECT id, nome, cpf FROM funcionario";  
  
$statement = $pdo->query($query);  
$statement->bindColumn(1, $id);  
$statement->bindColumn(2, $nome);  
$statement->bindColumn('cpf', $cpf);  
  
while ($row = $statement->fetch(PDO::FETCH_BOUND)) {  
    echo "$id:" . $nome . " " . $cpf . "</br>";  
}  
?
```



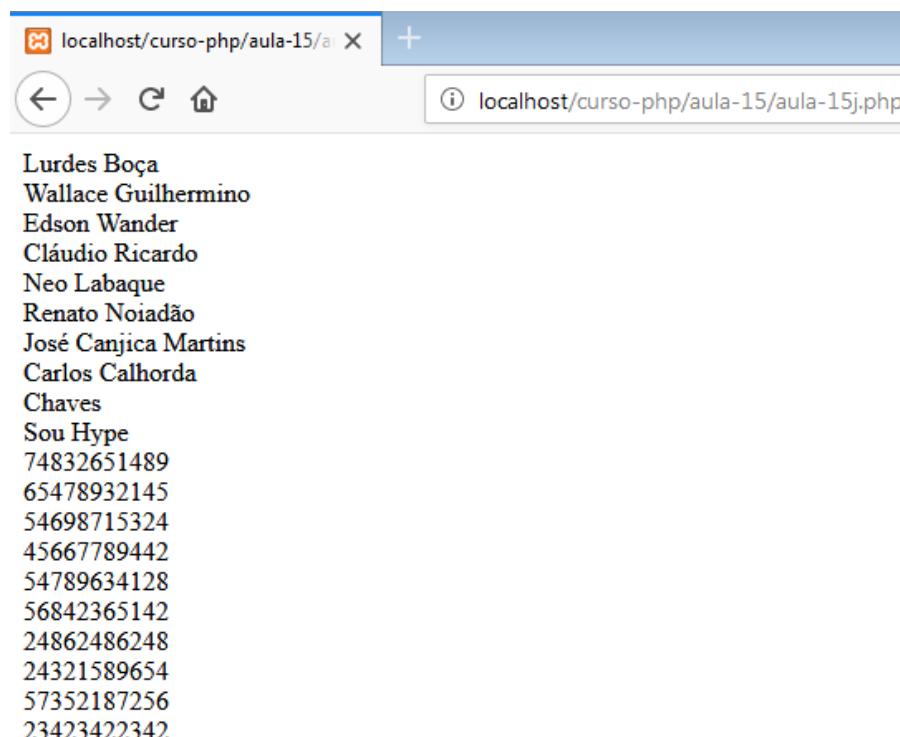
7:Lurdes Boça 74832651489  
8:Wallace Guilhermino 65478932145  
9:Edson Wander 54698715324  
10:Cláudio Ricardo 45667789442  
11:Neo Labaque 54789634128  
13:Renato Noiadão 56842365142  
14:José Canjica Martins 24862486248  
15:Carlos Calhorda 24321589654  
16:Chaves 57352187256  
17:Sou Hype 23423422342

## PDO::FETCH\_COLUMN (Estilos com argumentos)

PDO::FETCH\_COLUMN: retorna um array 2D onde os valores são os dados de uma única coluna e uma chave sequencial. Seu parâmetro é similar o fetchColumn, onde um inteiro é utilizado para indicar a coluna desejada.

### aula-15/aula-15j.php

```
<?php  
  
include_once "conexao.php";  
  
$statement = $pdo->query('SELECT nome, id FROM funcionario');  
$nomes = $statement->fetchAll(PDO::FETCH_COLUMN);  
  
foreach ($nomes as $nome) {  
    echo $nome."</br>";  
}  
  
$statement = $pdo->query('SELECT id, nome, cpf FROM funcionario');  
$cpfs = $statement->fetchAll(PDO::FETCH_COLUMN, 2);  
  
foreach ($cpfs as $cpf) {  
    echo $cpf."</br>";  
}
```

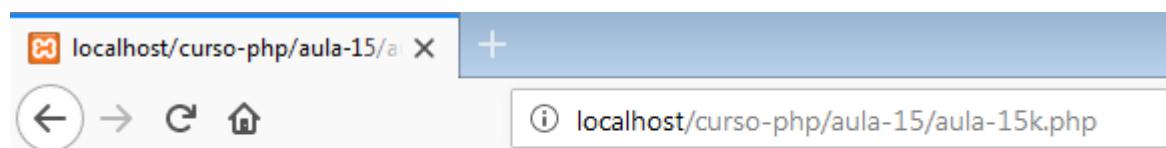


## PDO::FETCH\_FUNC (Estilos com argumentos)

PDO::FETCH\_FUNC: Permite que uma função/closure receberá em forma de parâmetros as colunas da query.

### aula-15/aula-15k.php

```
<?php  
  
include_once "conexao.php";  
  
$statement = $pdo->query('SELECT preco, titulo FROM livro');  
$livros = $statement->fetchAll(PDO::FETCH_FUNC, function($preco, $titulo){  
    $preco_no_cartao = round($preco + ($preco * 0.1),2);  
    return "$titulo: R$ {$preco} à vista e no cartão R$ {$preco_no_cartao}";  
});  
  
foreach ($livros as $preco) {  
    echo $preco."</br>";  
}
```



Sucesso na Vida: R\$ 39.99 à vista e no cartão R\$ 43.99  
Brincadeira em Excesso Virou Bobeira: R\$ 44.01 à vista e no cartão R\$ 48.41  
Fazendo Bolos com CakePHP: R\$ 89.95 à vista e no cartão R\$ 98.95  
Vamos Investigar?: R\$ 63.22 à vista e no cartão R\$ 69.54  
Portabilidade Manual: Um Tutorial Prático: R\$ 100.99 à vista e no cartão R\$ 111.09  
Brazil Mulambo: R\$ 9.99 à vista e no cartão R\$ 10.99  
Tudo pelo estudo: R\$ 1.99 à vista e no cartão R\$ 2.19  
Quem Cedo Madruga Deus Ajuda: R\$ 55.99 à vista e no cartão R\$ 61.59  
Madrugando: R\$ 18.89 à vista e no cartão R\$ 20.78  
Eu entendo seu conceito mas não concordo: R\$ 1.13 à vista e no cartão R\$ 1.24

## PDO e ORM (Object Relational Mapper)

- Alguns estilos do PDO permitem um mapeamento primitivo entre registros de uma tabela (modelo relacional) e o instâncias/objetos de uma classe (modelo orientado a objetos).
- As maiores limitações começam a surgir quando é necessário representar/mapear relacionamentos entre objetos.
- Para tarefas mais complexas de mapeamento é recomendável utilizar bibliotecas ORM de terceiros.

## PDO (Fetch Class)

O comportamento padrão do FETCH\_CLASS é chamar o construtor depois de colocar os valores nos atributos.

### aula-15/Livro.php

```
<?php
class Livro {
    public $id;
    public $titulo;
    private $edicao;
}
```

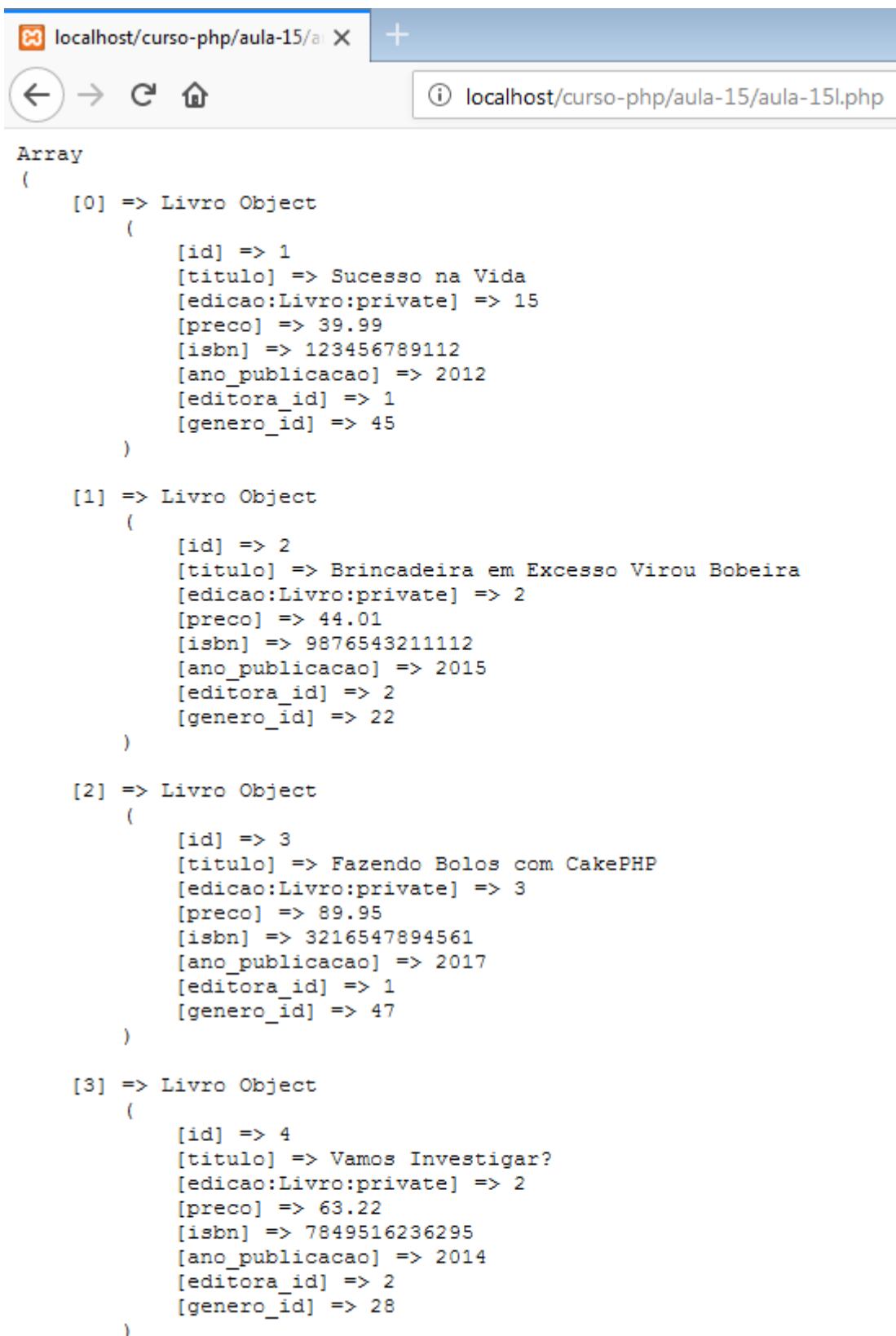
### aula-15/aula-15I.php

```
<?php

//recuperando dados
include_once "conexao.php";
include_once "Livro.php";

$stmt = $pdo->prepare("SELECT * FROM livro");
$stmt->execute();
$result = $stmt->fetchAll(\PDO::FETCH_CLASS, 'Livro');

print_r($result);
```



A screenshot of a web browser window. The address bar shows the URL `localhost/curso-php/aula-15/aula-15l.php`. The page content displays an array of four `Livro` objects, each represented as an associative array with properties like `id`, `titulo`, `preco`, etc.

```
Array
(
    [0] => Livro Object
        (
            [id] => 1
            [titulo] => Sucesso na Vida
            [edicao:Livro:private] => 15
            [preco] => 39.99
            [isbn] => 123456789112
            [ano_publicacao] => 2012
            [editora_id] => 1
            [genero_id] => 45
        )

    [1] => Livro Object
        (
            [id] => 2
            [titulo] => Brincadeira em Excesso Virou Bobeira
            [edicao:Livro:private] => 2
            [preco] => 44.01
            [isbn] => 9876543211112
            [ano_publicacao] => 2015
            [editora_id] => 2
            [genero_id] => 22
        )

    [2] => Livro Object
        (
            [id] => 3
            [titulo] => Fazendo Bolos com CakePHP
            [edicao:Livro:private] => 3
            [preco] => 89.95
            [isbn] => 3216547894561
            [ano_publicacao] => 2017
            [editora_id] => 1
            [genero_id] => 47
        )

    [3] => Livro Object
        (
            [id] => 4
            [titulo] => Vamos Investigar?
            [edicao:Livro:private] => 2
            [preco] => 63.22
            [isbn] => 7849516236295
            [ano_publicacao] => 2014
            [editora_id] => 2
            [genero_id] => 28
        )
)
```

```
[4] => Livro Object
(
    [id] => 5
    [titulo] => Portabilidade Manual: Um Tutorial Prático
    [edicao:Livro:private] => 2
    [preco] => 100.99
    [isbn] => 4568521597534
    [ano_publicacao] => 1997
    [editora_id] => 2
    [genero_id] => 18
)

[5] => Livro Object
(
    [id] => 6
    [titulo] => Brazil Mulambo
    [edicao:Livro:private] => 1
    [preco] => 9.99
    [isbn] => 1236547562111
    [ano_publicacao] => 2014
    [editora_id] => 1
    [genero_id] => 11
)

[6] => Livro Object
(
    [id] => 7
    [titulo] => Tudo pelo estudo
    [edicao:Livro:private] => 1
    [preco] => 1.99
    [isbn] => 12345678965
    [ano_publicacao] => 2002
    [editora_id] => 3
    [genero_id] => 23
)
```

```
[7] => Livro Object
(
    [id] => 8
    [titulo] => Quem Cedo Madruga Deus Ajuda
    [edicao:Livro:private] => 5
    [preco] => 55.99
    [isbn] => 9157357561
    [ano_publicacao] => 1997
    [editora_id] => 1
    [genero_id] => 13
)
[8] => Livro Object
(
    [id] => 9
    [titulo] => Madrugando
    [edicao:Livro:private] => 10
    [preco] => 18.89
    [isbn] => 5485315675165
    [ano_publicacao] => 1991
    [editora_id] => 4
    [genero_id] => 8
)
[9] => Livro Object
(
    [id] => 10
    [titulo] => Eu entendo seu conceito mas não concordo
    [edicao:Livro:private] => 1
    [preco] => 1.13
    [isbn] => 4534535325245
    [ano_publicacao] => 2017
    [editora_id] => 1
    [genero_id] => 21
)
)
```

## PDO::FETCH\_CLASS + PDO::FETCH\_CLASSTYPE

PDO::FETCH\_CLASSTYPE: Combinado com FETCH\_CLASS permite que a primeira coluna da query seja utilizada para definir qual classe utilizar para instância o registro. Pode ser muito útil para o caso de associações polimórficas com simulação de herança no modelo relacional.

### aula-15/aula-15m.php

```
<?php

include_once "conexao.php";

class Autor {}

class Funcionario {}

$stmt = $pdo->query(
    "SELECT 'Autor', nome FROM autor
    UNION
    SELECT 'Funcionario', nome FROM funcionario"
);

$objeto = $stmt->fetchAll(PDO::FETCH_CLASS | PDO::FETCH_CLASSTYPE);

echo "<pre>";
print_r($objeto);
echo "</pre>";
```



A screenshot of a web browser window. The address bar shows the URL `localhost/curso-php/aula-15/aula-15m.php`. The main content area displays the following PHP code output:

```
Array
(
    [0] => Autor Object
        (
            [nome] => Mosantos de Vilar dos Telles
        )

    [1] => Autor Object
        (
            [nome] => Jonas Guanabara da Silva
        )

    [2] => Autor Object
        (
            [nome] => Joselito de Cascatinha
        )

    [3] => Autor Object
        (
            [nome] => Luis Boça
        )

    [4] => Autor Object
        (
            [nome] => Charlinho Menino Guerreiro
        )

    [5] => Autor Object
        (
            [nome] => Dona Maxima
        )

    [6] => Autor Object
        (
            [nome] => Doutor Lincon
        )

    [7] => Autor Object
        (
            [nome] => Linhares
        )
)
```

```
[8] => Autor Object
(
    [nome] => Jonny Boganville
)

[9] => Autor Object
(
    [nome] => Jimmy Leroy
)

[10] => Autor Object
(
    [nome] => Professor Gilmar
)

[11] => Autor Object
(
    [nome] => Padre Quemedo
)

[12] => Autor Object
(
    [nome] => Lagreca
)

[13] => Autor Object
(
    [nome] => Dedé Carvoeiro
)

[14] => Autor Object
(
    [nome] => Carlos Carne
)

[15] => Autor Object
(
    [nome] => Seu Madruga
)
```

```
[16] => Autor Object
(
    [nome] => Teste 123
)

[17] => Funcionario Object
(
    [nome] => Lurdes Boça
)

[18] => Funcionario Object
(
    [nome] => Wallace Guilhermino
)

[19] => Funcionario Object
(
    [nome] => Edson Wander
)

[20] => Funcionario Object
(
    [nome] => Cláudio Ricardo
)

[21] => Funcionario Object
(
    [nome] => Neo Labaque
)

[22] => Funcionario Object
(
    [nome] => Renato Noiadão
)

[23] => Funcionario Object
(
    [nome] => José Canjica Martins
)

[24] => Funcionario Object
(
    [nome] => Carlos Calhorda
)

[25] => Funcionario Object
(
    [nome] => Chaves
)

[26] => Funcionario Object
(
    [nome] => Sou Hype
)

)
```

## **PDO::FETCH\_CLASS + PDO::FETCH\_PROPS\_LATE**

Podemos utilizar a flag `FETCH_PROPS_LATE` para que o PDO comerce a “colocar” os valores do objeto após a chamada do método construtor. Com isso podemos decorar nosso objeto e mapear um relacionamento 1..1, como neste exemplo.

### **aula-15/Funcionario.php**

```
<?php

class Funcionario {
    public $id;
    public $nome;
    public $habilitacao;

    public function __construct() {
        $this->habilitacao = new Habilitacao();
    }

    public function __set($name, $value) {
        if (array_key_exists($name, get_object_vars($this->habilitacao))) {
            $this->habilitacao->$name = $value;
        } else {
            $this->$name = $value;
        }
    }
}
```

### **aula-15/Habilitacao.php**

```
<?php

class Habilitacao {
    public $numero;
    public $categoria;
}
```

## **aula-15/aula-15n.php**

```
<?php

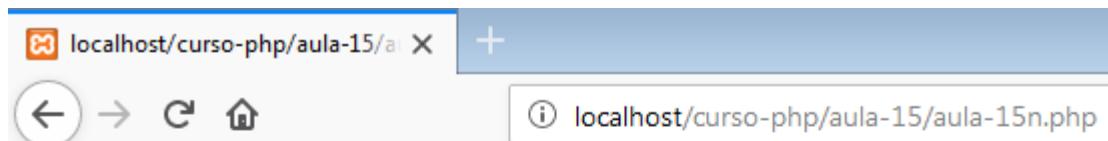
include 'conexao.php';
include 'Funcionario.php';
include 'Habilitacao.php';

$statement = $pdo->prepare(
    "SELECT * FROM funcionario "
    . "LEFT JOIN habilitacao "
    . "ON funcionario.id = habilitacao.funcionario_id"
);

$statement->execute();

$funcionarios = $statement->fetchAll(\PDO::FETCH_CLASS |
\PDO::FETCH_PROPS_LATE, Funcionario::class);

foreach ($funcionarios as $funcionario) {
    echo "{$funcionario->nome} com Habilidade nº {$funcionario->habilitacao-
>numero} <br/>";
}
```



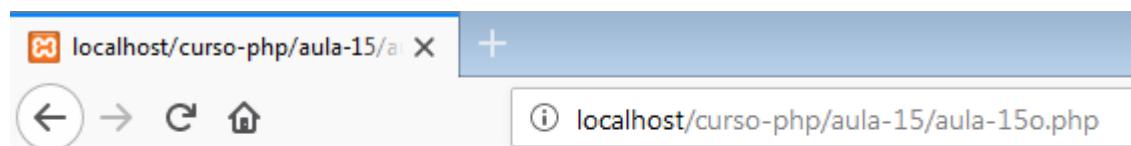
Lurdes Boça com Habilidade nº 78495162354  
Wallace Guilhermino com Habilidade nº  
Edson Wander com Habilidade nº  
Cláudio Ricardo com Habilidade nº 99885523654  
Neo Labaque com Habilidade nº 12396348525  
Renato Noiadão com Habilidade nº 45687512598  
José Canjica Martins com Habilidade nº 21575698423  
Carlos Calhorda com Habilidade nº 14785236548  
Chaves com Habilidade nº 23484562848  
Sou Hype com Habilidade nº 44334345345

## PDO::FETCH\_OBJ

PDO::FETCH\_OBJ: Retorna os registros em forma de objetos genéricos (instâncias de StdClass).

### aula-15/aula-15o.php

```
<?php  
  
include_once "conexao.php";  
  
$statement = $pdo->query('SELECT * FROM livro');  
$livros = $statement->fetchall(PDO::FETCH_OBJ);  
  
foreach ($livros as $livro) {  
    echo "instância de ".get_class($livro).":";  
    echo $livro->título."<br>";  
}
```



instância de stdClass:Sucesso na Vida  
instância de stdClass:Brincadeira em Excesso Virou Bobeira  
instância de stdClass:Fazendo Bolos com CakePHP  
instância de stdClass:Vamos Investigar?  
instância de stdClass:Portabilidade Manual: Um Tutorial Prático  
instância de stdClass:Brazil Mulambo  
instância de stdClass:Tudo pelo estudo  
instância de stdClass:Quem Cedo Madruga Deus Ajuda  
instância de stdClass:Madrugando  
instância de stdClass:Eu entendo seu conceito mas não concordo

## fetchObject

Vale a pena salientar que o `fetchObject` cria a instância e seta os valores antes mesmo do método construtor ser executado.

### aula-15/aula-15p.php

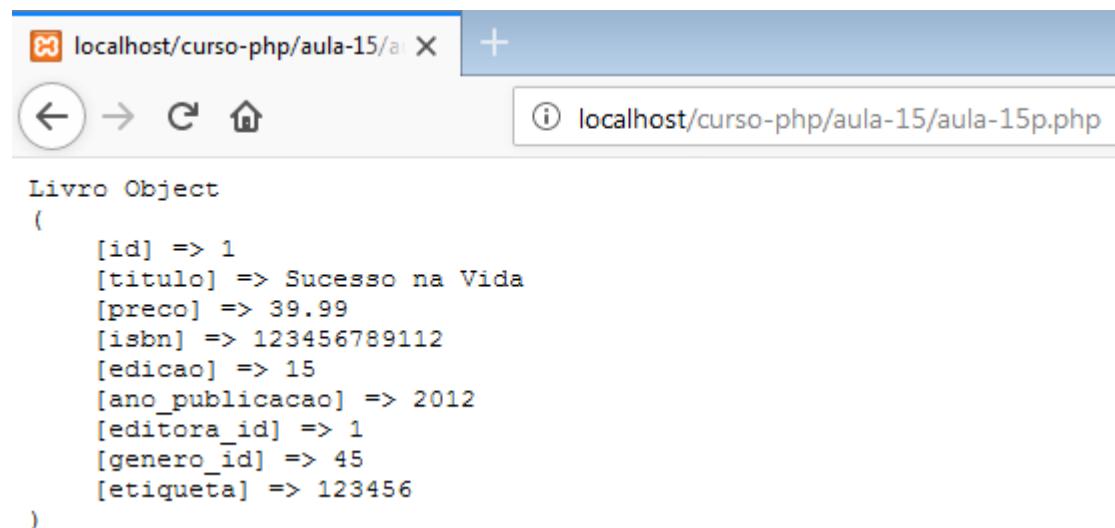
```
<?php

include 'conexao.php';

class Livro {
    public function __construct($etiqueta = null) {
        $this->etiqueta = $etiqueta;
    }
}

$stmt = $pdo->query("SELECT * FROM livro");
$livro = $stmt->fetchObject(Livro::class, [123456]);

echo "<pre>";
print_r($livro);
echo "</pre>";
```



```
Livro Object
(
    [id] => 1
    [titulo] => Sucesso na Vida
    [preco] => 39.99
    [isbn] => 123456789112
    [edicao] => 15
    [ano_publicacao] => 2012
    [editora_id] => 1
    [genero_id] => 45
    [etiqueta] => 123456
)
```

## PDO::FETCH\_INTO

PDO::FETCH\_INTO: Permite atualizar uma instância existente da classe solicitada, mapeando as colunas do conjunto da query para os atributos nomeadas na classe.

Atenção: Diferente do FETCH\_CLASS, o FETCH\_INTO não é capaz de modificar valores de atributos privados ou protegidos

### aula-15/aula-15q.php

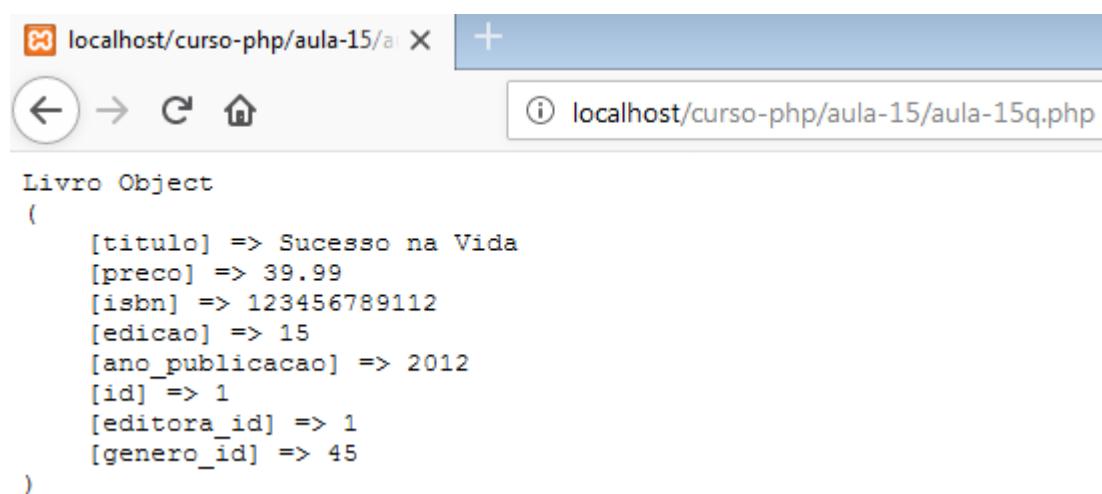
```
<?php

include 'conexao.php';

class Livro {
    public $titulo;
    public $preco;
    public $isbn;
    public $edicao;
    public $ano_publicacao;
}

$livro = new Livro();
$stmt = $pdo->query('SELECT * FROM livro');
$stmt->setFetchMode(PDO::FETCH_INTO, $livro);
$stmt->fetch();

echo "<pre>";
print_r($livro);
echo "</pre>";
```



```
Livro Object
(
    [titulo] => Sucesso na Vida
    [preco] => 39.99
    [isbn] => 123456789112
    [edicao] => 15
    [ano_publicacao] => 2012
    [id] => 1
    [editora_id] => 1
    [genero_id] => 45
)
```

## **PDO::FETCH\_CLASS + PDO::FETCH\_SERIALIZE**

PDO::FETCH\_SERIALIZE: Tem efeito similar ao FETCH\_INTO porém é utiliza os comportamentos da serialização (interface) nos objetos.

### **aula-15/aula-15r.php**

```
<?php

include 'conexao.php';

class Log implements \Serializable {
    public $id;
    public $dados;
    public function serialize() {
        return serialize((array) $this);
    }

    public function unserialize($serialized): void {
        foreach (unserialize($serialized) as $p => $v) {
            $this->{$p} = $v;
        }
    }
}

$stmt = $pdo->query('SELECT dados FROM log');
$stmt->setFetchMode(PDO::FETCH_CLASS|PDO::FETCH_SERIALIZE,
Log::class);
$logs = $stmt->fetchAll();

foreach ($logs as $log) {
    echo "id: $log->id | ";
    echo "dados: " . var_export($log->dados, true);
    echo "<br/>";
}
```

## PDO (operações com cursor)

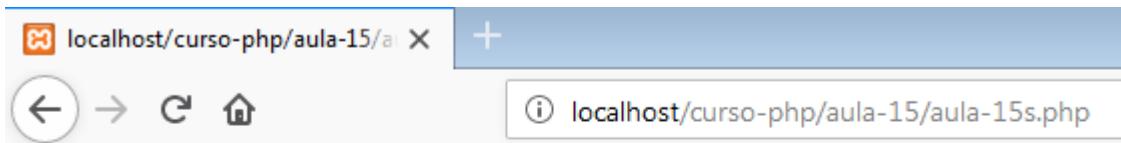
- Quando um cursor é criado para uma consulta, é possível iterar sobre o conjunto de linhas sem obter o resultado inteiro da mesma de uma só vez.
- Com um cursor não rolável (forward-only) é possível efetuar FETCH em cada linha no máximo uma vez, e o cursor se move automaticamente para a linha seguinte.
- Já os cursores roláveis (scrollable) permitem iterar o resultado em diversas direções, inclusive para trás e por isso podem acessar a mesma linha no conjunto do resultados várias vezes. Assim, modificações de dados (inserir, atualizar, excluir operações) de outras transações podem ter um impacto no conjunto de resultados.

## PDO Cursores (Exemplo Pedidos)

Em SGBDs que suportam o recurso de cursor scroll, é possível, no momento do método fetch(), posicionar o cursor em diferentes pontos do resultado, bem como movimentá-lo para frente ou para trás. No SQL Server o ABS começa com zero.

### aula-15/aula-15s.php

```
<?php
include_once '../conexao_postgres.php'; //sqlserver, oracle e postgres
$query = "select * from pedido order by id";
$statement = $pdo->prepare($query, [PDO::ATTR_CURSOR =>
PDO::CURSOR_SCROLL]);
$statement->execute();
$pedido = $statement->fetch(PDO::FETCH_LAZY, PDO::FETCH_ORI_FIRST);
print "Primeiro pedido (id:{$pedido['id']}) ocorreu em : {$pedido['data']}<br/>";
$pedido = $statement->fetch(PDO::FETCH_LAZY, PDO::FETCH_ORI_NEXT);
print "Próximo pedido (id:{$pedido['id']}) ocorreu em : {$pedido['data']}<br/>";
$pedido = $statement->fetch(PDO::FETCH_LAZY, PDO::FETCH_ORI_LAST);
print "Último pedido (id:{$pedido['id']}) ocorreu em: {$pedido['data']}<br/>";
$pedido = $statement->fetch(PDO::FETCH_LAZY, PDO::FETCH_ORI_PRIOR);
print "Penúltimo pedido (id:{$pedido['id']}) ocorreu em: {$pedido['data']}<br/>";
$pedido = $statement->fetch(PDO::FETCH_LAZY, PDO::FETCH_ORI_ABS, 3);
print "Terceiro pedido (id:{$pedido['id']}) em: {$pedido['data']}<br/>";
```



Primeiro pedido (id:1) ocorreu em : 2016-03-01 00:00:00  
Próximo pedido (id:2) ocorreu em : 2014-12-11 13:00:00  
Último pedido (id:38) ocorreu em: 2019-03-24 00:00:00  
Penúltimo pedido (id:37) ocorreu em: 2019-03-24 00:00:00  
Terceiro pedido (id:3) em: 2017-03-30 06:30:37

# Aula 16 - Transaction, ACID, PDOException, LastInsertId

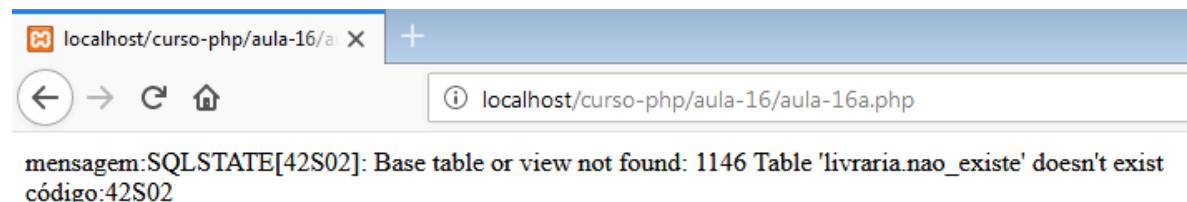
## PDOException

O PDO também faz parte da Taxonomia de Throwable e possui uma Exception com informações diretamente específicas do SGBD que podem auxiliar no tratamento do erro.

Atenção: Para que o PDO lance exceções, é necessário utilizar o atributo de conexão PDO::ATTR\_ERRMODE com o valor PDO::ERRMODE\_EXCEPTION.

### aula-16/aula-16a.php

```
<?php  
  
include_once 'conexao.php';  
  
try{  
    $pdo->query('SELECT * FROM nao_existe');  
} catch (\PDOException $t) {  
    echo "mensagem:". $t->getMessage(). "<br/>". "código:". $t->getCode();  
}
```



## SQL DML: Inserção, Atualização e Remoção

- Os recursos da linguagem SQL são normalmente divididos em conjuntos onde cada um destes possui um propósito específico.
- Um destes conjuntos é chamado de DML (Data Manipulation Language / Linguagem de Manipulação de Dados).
- O DML é composto pelas seguintes instruções (statements): INSERT, UPDATE e DELETE.
- É fortemente recomendado utilizar prepared statement ao executar qualquer instrução DML no PDO.

### PDO (Inserção)

A instrução (statement) INSERT SQL permite adicionar um ou mais registros a qualquer tabela única em um banco de dados relacional.

#### **aula-16/aula-16b.php**

```
<?php

include_once "conexao.php";

$nome = "Geraldo";
$cpf = "1234";

try {
    $statement = $pdo->prepare("INSERT INTO funcionario (nome, cpf)"
    . "VALUES (?,?)");
    $cadastro = $statement->execute([$nome, $cpf]);

    if($cadastro){
        echo "Funcionário cadastrado com sucesso!";
    }
}

} catch (\PDOException $t) {
    echo "mensagem:".$t->getMessage()."<br/>".
    "código:".$t->getCode();
}
```

localhost/curso-php/aula-16/a... X +

← → ⌂ ⌂

localhost/curso-php/aula-16/aula-16b.php

Funcionário cadastrado com sucesso!

A mostrar registos de 0 - 10 (11 total, A consulta demorou 0,0000 segundos.)

SELECT \* FROM `funcionario`

Mostrar tudo | Número de registos: 50 ▾ Filtrar registos: Pesquisar esta tabela

+ Opções

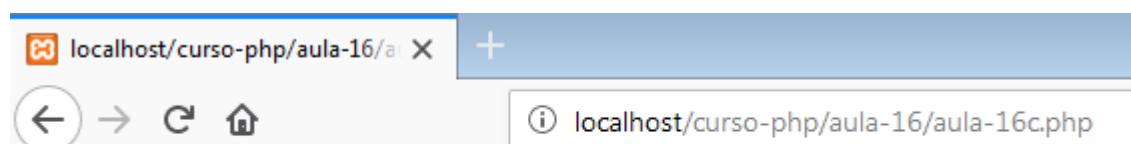
		id	nome	cpf	gerente_id
<input type="checkbox"/>	Edita  Copiar  Apagar	7	Lurdes Boça	74832651489	NULL
<input type="checkbox"/>	Edita  Copiar  Apagar	8	Wallace Guilhermino	65478932145	7
<input type="checkbox"/>	Edita  Copiar  Apagar	9	Edson Wander	54698715324	8
<input type="checkbox"/>	Edita  Copiar  Apagar	10	Cláudio Ricardo	45667789442	7
<input type="checkbox"/>	Edita  Copiar  Apagar	11	Neo Labaque	54789634128	7
<input type="checkbox"/>	Edita  Copiar  Apagar	13	Renato Noidão	56842365142	10
<input type="checkbox"/>	Edita  Copiar  Apagar	14	José Canjica Martins	24862486248	7
<input type="checkbox"/>	Edita  Copiar  Apagar	15	Carlos Calhorda	24321589654	8
<input type="checkbox"/>	Edita  Copiar  Apagar	16	Chaves	57352187256	NULL
<input type="checkbox"/>	Edita  Copiar  Apagar	17	Sou Hype	23423422342	NULL
<input type="checkbox"/>	Edita  Copiar  Apagar	18	Geraldo	1234	NULL

## PDO (Atualização)

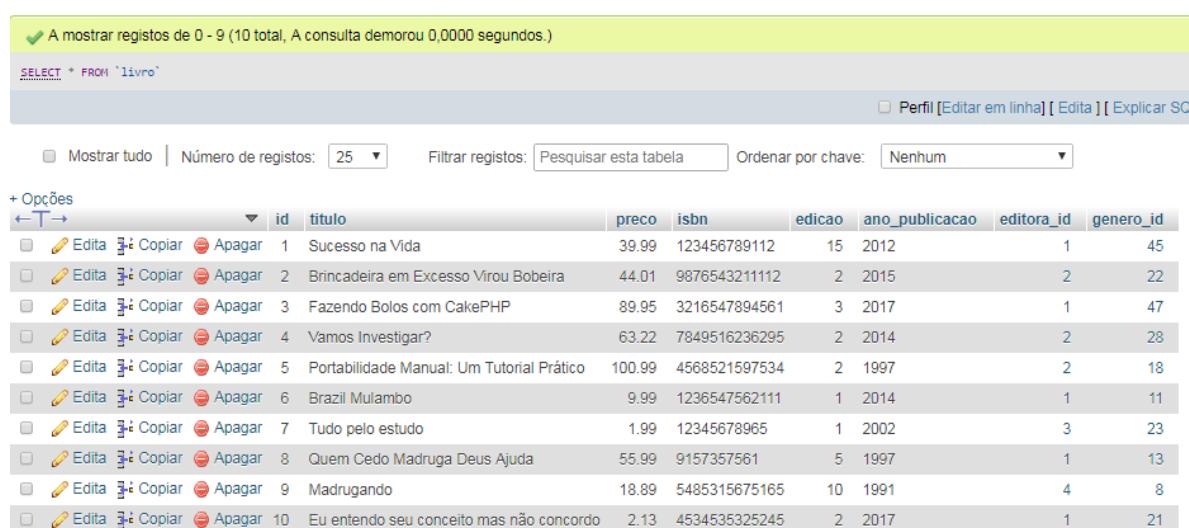
A instrução (statement) UPDATE SQL permite alterar os dados de um ou mais registros em uma tabela. Todas as linhas podem ser atualizadas ou um subconjunto pode ser escolhido usando uma condição (WHERE).

### aula-16/aula-16c.php

```
<?php  
  
include_once "conexao.php";  
  
$id = 10;  
  
try {  
    $stmt = $pdo->prepare("UPDATE livro SET preco = preco + 1, edicao = 2  
WHERE id = :id");  
    $update = $stmt->execute(["id" => $id]);  
    if($update){  
        echo "Livro atualizado com sucesso";  
    }  
  
} catch (\PDOException $t) {  
    echo "mensagem:".$t->getMessage()."  
"código:".$t->getCode();  
}
```



Livro atualizado com sucesso



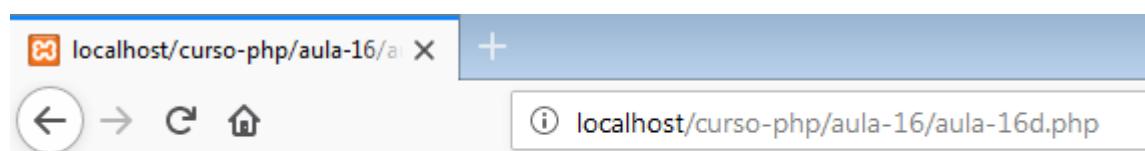
A mostrar registos de 0 - 9 (10 total, A consulta demorou 0,0000 segundos.)								
SELECT * FROM `livro`								
Perfil [Editar em linha] [ Edita ] [ Explicar SC ]								
Mostrar tudo   Número de registo: 25 ▾			Filtrar registo: Pesquisar esta tabela			Ordenar por chave: Nenhum ▾		
+ Opcões	Edita	Copiar	Apagar	id	titulo	preco	isbn	edicao
<input type="checkbox"/>				1	Sucesso na Vida	39.99	123456789112	15
<input type="checkbox"/>				2	Brincadeira em Excesso Virou Bobeira	44.01	9876543211112	2
<input type="checkbox"/>				3	Fazendo Bolos com CakePHP	89.95	3216547894561	3
<input type="checkbox"/>				4	Vamos Investigar?	63.22	7849516236295	2
<input type="checkbox"/>				5	Portabilidade Manual: Um Tutorial Prático	100.99	4568521597534	2
<input type="checkbox"/>				6	Brazil Mulambo	9.99	1236547562111	1
<input type="checkbox"/>				7	Tudo pelo estudo	1.99	12345678965	1
<input type="checkbox"/>				8	Quem Cedo Madruga Deus Ajuda	55.99	9157357561	5
<input type="checkbox"/>				9	Madrugando	18.89	5485315675165	10
<input type="checkbox"/>				10	Eu entendo seu conceito mas não concordo	2.13	4534535325245	2

## PDO (Remoção)

A instrução (statement) DELETE SQL permite remover um ou mais registros de uma tabela. Um subconjunto pode ser definido para exclusão usando uma condição (WHERE), caso contrário, todos os registros serão removidos. Alguns SGBDs, como o MySQL, permitem a exclusão de linhas de várias tabelas com uma instrução DELETE (multi-table delete)

### aula-16/aula-16d.php

```
<?php  
  
include_once "conexao.php";  
  
$id = 18;  
  
try {  
    $stmt = $pdo->prepare("DELETE FROM funcionario WHERE id = :id");  
    $delete = $stmt->execute(['id' => $id]);  
    if($delete){  
        echo "Funcionário excluído com sucesso!";  
    }  
} catch (\PDOException $t) {  
    echo "mensagem:".$t->getMessage()."<br/>".  
    "código:".$t->getCode();  
}
```



A mostrar registos de 0 - 9 (10 total, A consulta demorou 0,0000 segundos.)

```
SELECT * FROM `funcionario`
```

Mostrar tudo | Número de registos: 50 | Filtrar registos:

+ Opções

<input type="checkbox"/>	Edita	Copiar	Apagar	7	Lurdes Boça	74832651489
<input type="checkbox"/>	Edita	Copiar	Apagar	8	Wallace Guilhermino	65478932145
<input type="checkbox"/>	Edita	Copiar	Apagar	9	Edson Wander	54698715324
<input type="checkbox"/>	Edita	Copiar	Apagar	10	Cláudio Ricardo	45667789442
<input type="checkbox"/>	Edita	Copiar	Apagar	11	Neo Labaque	54789634128
<input type="checkbox"/>	Edita	Copiar	Apagar	13	Renato Noiadão	56842365142
<input type="checkbox"/>	Edita	Copiar	Apagar	14	José Canjica Martins	24862486248
<input type="checkbox"/>	Edita	Copiar	Apagar	15	Carlos Calhorda	24321589654
<input type="checkbox"/>	Edita	Copiar	Apagar	16	Chaves	57352187256
<input type="checkbox"/>	Edita	Copiar	Apagar	17	Sou Hype	23423422342

## Conjunto de DMLs (sem transação)

É muito comum a utilização de duas ou mais instruções SQL DML de forma sequencial para cumprir a lógica de um caso de uso da aplicação.

Um exemplo seria o caso de uso: “transferência bancária”

No exemplo a seguir, R\$ 200 serão “transferidos” da conta do cliente 1 para o cliente 2.

+ Opções

<input type="checkbox"/>	Edita	Copiar	Apagar	1	1000.00	1
<input type="checkbox"/>	Edita	Copiar	Apagar	2	200.00	2

## **aula-16/aula-16e.php**

```
<?php

include_once "conexao.php";

try {
    $stmt1 = $pdo->prepare("UPDATE conta_corrente SET saldo = saldo - 200
WHERE cliente_id = 1");
    $stmt1->execute();
    sleep(25); // se durante este tempo o SGBD cair, a próxima instrução nunca ocorrerá
    $stmt2 = $pdo->prepare("UPDATE conta_corrente SET saldo = saldo + 200
WHERE cliente_id = 2");
    $stmt2->execute();
} catch (\PDOException $t) {
    echo "mensagem:". $t->getMessage(). "<br/>".
    "código:". $t->getCode();
}
```

The screenshot shows a database management interface with the following details:

- Message Bar:** A green bar at the top indicates: "A mostrar registos de 0 - 1 (2 total, A consulta demorou 0,0000)".
- SQL Query:** The query "SELECT \* FROM `conta\_corrente`" is displayed.
- Table View:** The "conta\_corrente" table has the following data:

	id	saldo	cliente_id
<input type="checkbox"/>	1	800.00	1
<input type="checkbox"/>	2	200.00	2
- Toolbar:** Includes buttons for "Mostrar tudo" (Show all), "Número de registo" (Number of records) set to 25, and "Filtrar" (Filter).
- Options:** A "+ Opções" button is visible.

## **PDO x Transação**

- Uma transação, no contexto de um banco de dados, é uma unidade lógica que é executada de forma independente para recuperação ou atualização de dados.
- No SQL ANSI uma transação começa com BEGIN TRANSACTION e, após a mesma, todas as instruções serão parte desta transação.
- Para “confirmar” a transação, faz-se necessário utilizar a instrução COMMIT
- Para “cancelar” todas as instruções da transação e retornar o banco ao estado original, faz-se necessário utilizar o comando ROLLBACK
- Em bancos de dados relacionais, as transações do banco de dados devem ser atômicas, consistentes, isoladas e duráveis - resumidas como o acrônimo ACID.
- Alguns SGBDs como o MySQL, utiliza outra sintaxe (START ao invés de BEGIN). Utilizando o PDO estas diferenças são suprimidas pois o mesmo possui um método único para todos.

## **PDO Transaction (Inserção)**

No próximo exemplo, dois inserts são executados dentro de uma transação, porém, apenas com a confirmação feita com o commit, é que as mesmas serão de fato executadas.

Para que a transação não fique aberta em caso de erro, utilizaremos o rollback dentro do catch

## **aula-16/aula-16f.php**

```
<?php

include_once "conexao.php";

try {
    $pdo->beginTransaction();
    $stmt1 = $pdo->prepare("UPDATE conta_corrente SET saldo = saldo - 200
WHERE cliente_id = 1");
    $stmt1->execute();
    sleep(25);
    $stmt2 = $pdo->prepare("UPDATE conta_corrente SET saldo = saldo + 200
WHERE cliente_id = 2");
    $stmt2->execute();
    $pdo->commit();
} catch (\PDOException $t) {
    $pdo->rollback();
    echo "mensagem:".$t->getMessage()."<br/>".
    "código:".$t->getCode();
}
```

			<b>id</b>	<b>saldo</b>	<b>cliente_id</b>
<input type="checkbox"/>	 Edita	 Copiar	 Apagar	1	1000.00
<input type="checkbox"/>	 Edita	 Copiar	 Apagar	2	200.00

## **Transação ACID**

- Atomicidade: Atomicidade garante que cada transação seja tratada de forma "unitária" e, se alguma das instruções que constituem uma transação não for concluída, a transação inteira falhará e o banco de dados permanecerá inalterado;
- Consistência: A consistência garante que uma transação só pode trazer o banco de dados de um estado válido para outro, mantendo as regras do banco: quaisquer dados gravados no banco de dados devem ser válidos de acordo com todas as regras definidas, incluindo restrições, triggers e qualquer combinação dos mesmos. Isso evita corrupção do banco de dados por uma transação ilegal, mas não garante que uma transação esteja correta;
- Isolamento: As transações geralmente são executadas simultaneamente (por exemplo, ler e gravar em várias tabelas ao mesmo tempo). O isolamento garante que a execução simultânea de transações deixe o banco de dados no mesmo estado que teria sido obtido se as transações fossem executadas sequencialmente. O isolamento é o principal objetivo do controle de concorrência;
- Durabilidade: A durabilidade garante que uma vez que uma transação tenha sido confirmada, ela permanecerá comprometida mesmo no caso de uma falha no sistema (por exemplo, falta de energia ou falha). Isso geralmente significa que as transações concluídas (ou seus efeitos) são registradas na memória não volátil.

## **PDO Transaction (Inserção)**

No próximo exemplo, dois inserts são executados dentro de uma transação, porém, apenas com a confirmação feita com o commit, é que as mesmas serão de fato executadas.

Para que a transação não fique aberta em caso de erro, utilizaremos o rollback dentro do catch.

## PDO::lastInsertId

### aula-16/aula-16g.php

```
<?php

include_once "conexao.php";

try {
    $pdo->beginTransaction();
    $sql = "INSERT INTO funcionario (nome, cpf) VALUES(:nome,:cpf)";
    $driver = $pdo->getAttribute(PDO::ATTR_DRIVER_NAME);

    if($driver=='oci'){
        $sql .= 'RETURNING id INTO :last_id';
    }

    $statement = $pdo->prepare($sql);
    $statement->bindValue('nome','Jaqueline');
    $statement->bindValue('cpf','12269736044');

    if($driver == 'oci'){
        $statement->bindParam('last_id', $funcionario_id, PDO::PARAM_INT, 8);
    }

    $statement->execute();

    if($driver != 'oci'){
        $funcionario_id = $pdo->lastInsertId();
    }

    $sql2 = "INSERT INTO habilitacao (numero, categoria, funcionario_id)
VALUES(:numero,:categoria,:funcionario_id) ";
    $statement2 = $pdo->prepare($sql2);
    $statement2->execute(['95685512398','B',$funcionario_id]);

    $pdo->commit();
}catch (Exception $e){
    $pdo->rollback();
    throw $e;
}
```

## tabela funcionario

A mostrar registos de 0 - 10 (11 total, A consulta demorou 0,0000 segundos.)

```
SELECT * FROM `funcionario`
```

Mostrar tudo | Número de registos: 25 ▾ Filtrar registos: Pesquisar esta tāb

+ Opções

	Edita	Copiar	Apagar	id	nome	cpf	gerente_id
<input type="checkbox"/>				7	Lurdes Boça	74832651489	NULL
<input type="checkbox"/>				8	Wallace Guilhermino	65478932145	7
<input type="checkbox"/>				9	Edson Wander	54698715324	8
<input type="checkbox"/>				10	Cláudio Ricardo	45667789442	7
<input type="checkbox"/>				11	Neo Labaque	54789634128	7
<input type="checkbox"/>				13	Renato Noiadão	56842365142	10
<input type="checkbox"/>				14	José Canjica Martins	24862486248	7
<input type="checkbox"/>				15	Carlos Calhorda	24321589654	8
<input type="checkbox"/>				16	Chaves	57352187256	NULL
<input type="checkbox"/>				17	Sou Hype	23423422342	NULL
<input type="checkbox"/>				19	Jaqueline	12269736044	NULL

## tabela habilitacao

	Edita	Copiar	Apagar	id	numero	categoria	funcionario_id
<input type="checkbox"/>				1	78495162354	B	7
<input type="checkbox"/>				2	99885523654	AD	10
<input type="checkbox"/>				3	45687512598	C	13
<input type="checkbox"/>				5	12396348525	A	11
<input type="checkbox"/>				6	21575698423	AB	14
<input type="checkbox"/>				7	14785236548	C	15
<input type="checkbox"/>				8	23484562848	AB	16
<input type="checkbox"/>				9	44334345345	C	17
<input type="checkbox"/>				10	95685512398	B	19

# Aula 17 - PDO X BLOB MySQL, Postgres, SQLServer e Oracle Bytea, Varbinary

## PDO x BLOB

- A maioria dos SGDS tem suporte ao armazenamento de BLOBs
- Um Binary Large Object (BLOB) é uma coleção de dados binários armazenados como uma entidade única em um sistema de gerenciamento de banco de dados. Os blobs são tipicamente imagens, áudio ou outros objetos multimídia, embora às vezes o código executável binário seja armazenado como um blob.
- Com o PDO o bind (vinculação) de um valor BLOB é feito utilizando a tipagem como LOB através do PARAM\_LOB
- Alguns bancos como o SQL Server exigem o uso de um constante proprietária para indicar corretamente ao Drive ODBC o uso recurso.
- Outros bancos como o Oracle exigem a necessidade da criação de um BLOB vazio via SGBD e, através de um ponteiro virtual de arquivo (resource) a “persistência” dos dados.

## PDO: Inserção de Binário (formulário)

Neste exemplo, será submetido via formulário o id do funcionário e uma imagem que será persistida na tabela foto.

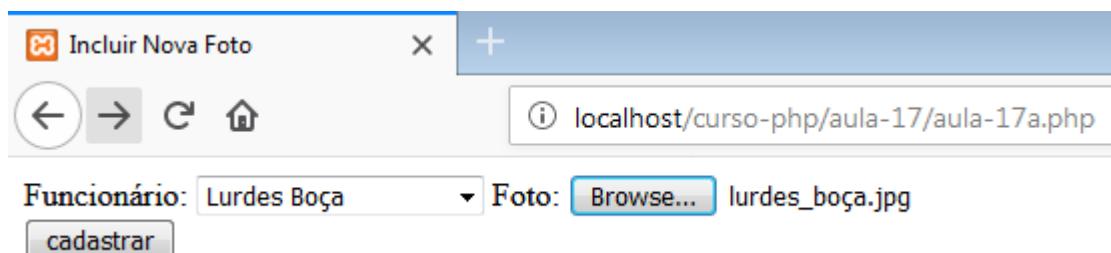
#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Comentários	Extra	Acções
1	id	int(11)			Não	None		AUTO_INCREMENT	Muda  Elimina ▾ Mais
2	binario	blob			Não	None			Muda  Elimina ▾ Mais
3	mimetype	varchar(255)	utf8_unicode_ci		Não	None			Muda  Elimina ▾ Mais
4	funcionario_id	int(11)			Não	None			Muda  Elimina ▾ Mais

## **aula-17/aula-17a.php**

```
<!DOCTYPE html>
<html>
    <head>
        <title>Incluir Nova Foto</title>
        <meta charset="utf-8"/>
    </head>
    <body>
        <form action="blob_action.php" method="post" enctype="multipart/form-data">
            Funcionário:
            <select name="funcionario_id">
                <?php include 'conexao.php';
                    $funcionarios = $pdo->query('SELECT id, nome FROM funcionario')-
                >fetchAll(PDO::FETCH_KEY_PAIR);
                foreach($funcionarios as $key => $value){
                    echo "<option value='$key'>$value</option>";
                }
            ?>
            </select>

            Foto:
            <input type="file" name="foto" accept="image/*"><br>

            <input type="submit" value="cadastrar"/>
        </form>
    </body>
</html>
```



## **aula-17/blob\_action.php**

```
<?php

include 'conexao.php';

try {
    $pdo->beginTransaction();
    $funcionario_id = $_REQUEST['funcionario_id'];
    $arquivo = $_FILES['foto']['tmp_name'];
    $binario = file_get_contents($arquivo);
    $mimetype = $_FILES['foto']['type'];
    $sql = "INSERT INTO foto (binario, mimetype, funcionario_id) ";
    $sql_values = " VALUES (:binario, :mimetype, :funcionario_id)";
    $driver = $pdo->getAttribute(PDO::ATTR_DRIVER_NAME);

    if ($driver == 'oci') {
        $sql_values = " VALUES (EMPTY_BLOB(), :mimetype, :funcionario_id)"
            . " RETURNING binario INTO :binario";
    }

    $statement = $pdo->prepare($sql . $sql_values);

    switch ($driver) {
        case 'sqlsrv':
            $statement->bindParam('binario', $binario, PDO::PARAM_LOB, 0,
PDO::SQLSRV_ENCODING_BINARY);
            break;
        case 'oci':
            $statement->bindParam('binario', $binario_resource, PDO::PARAM_LOB);
            $binario_resource = fopen($arquivo, 'rb');
            break;
        default:
            $statement->bindParam('binario', $binario, PDO::PARAM_LOB);
            break;
    }

    $statement->bindValue('mimetype', $mimetype, PDO::PARAM_STR);
    $statement->bindValue('funcionario_id', $funcionario_id, PDO::PARAM_INT);
    $statement->execute();
    $pdo->commit();
} catch (\PDOException $ex) {
    $pdo->rollback();
    echo $ex->getMessage();
}
```

A mostrar registos de 0 - 0 (1 total, A consulta demorou 0,0000 segundos.)

```
SELECT * FROM `foto`
```

Mostrar tudo | Número de registos: 25 | Filtrar registos: Pesquisar esta t

+ Opções

← → | id | binario | mimetype | funcionario\_id

Edita  Copiar  Apagar | 1 | [BLOB - 28.6 KB] | image/jpeg | 7

## Tipo Resource

- Um resource é uma variável especial, mantendo uma referência a um recurso externo. Recursos são criados e usados por funções especiais.
- Resources normalmente funcionam como identificadores (referências) especiais para arquivos abertos, conexões de banco de dados, áreas de tela de imagem e semelhantes.
- Devido a natureza dos resources, a conversão dos mesmos não é possível.

## Recuperação de binários do Banco (blob/varbinary)

Neste exemplo recuperamos uma imagem armazenada no banco. Alguns drivers utilizam um link em forma de resource (recurso) para acessar o binário de forma otimizada. Para estes casos, utilizamos o stream\_get\_contents() para recuperar o conteúdo binário.

ob\_clean() é uma função do PHP para ajudar a limpar o “output buffer” e evitar algum caráter indesejável na composição do echo do binário, o que poderia corromper o mesmo

## **aula-17/carrega\_foto.php**

```
<?php

include 'conexao.php';

$id = $_REQUEST['funcionario_id'];

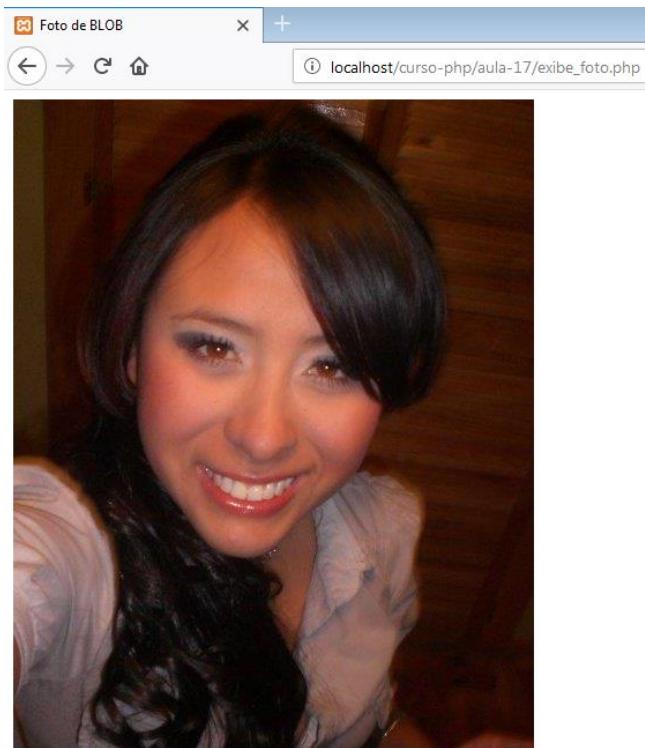
try {
    $statement = $pdo->prepare("SELECT * FROM foto WHERE funcionario_id = ?");
    $statement->execute([$id]);
    $foto = $statement->fetch(PDO::FETCH_ASSOC);
} catch (Exception $exc) {
    echo $exc->getTraceAsString();
}

ob_clean();
header("Content-type: {$foto['mimetype']}");

if (is_resource($foto['binario'])) {
    echo stream_get_contents($foto['binario']);
} else {
    echo ($foto['binario']);
}
```

## **aula-17/exibe\_foto.php**

```
<!DOCTYPE html>
<html lang="pt-BR">
    <head>
        <title>Foto de BLOB</title>
        <meta charset="UTF-8">
    </head>
    <body>
        
    </body>
</html>
```



## **aula-17/lista\_funcionarios.php**

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>Lista de funcionários (com foto)</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <?php
      include 'conexao.php';

      echo "<table border>";
      echo "<tr><th>Nome</th><th>Foto</th></tr>";
      $funcionarios = $pdo->query("SELECT * FROM funcionario")->fetchAll();

      foreach($funcionarios as $funcionario){
        echo "<tr>";
        echo "<td>{$funcionario['nome']}
```



A screenshot of a web browser window titled "Lista de funcionários (com foto)". The address bar shows the URL "localhost/curso-php/aula-17/lista\_funcionarios.php". The page displays a table with two columns: "Nome" and "Foto". The first row contains the name "Lurdes Boça" in the "Nome" column and a photograph of a woman with dark hair and a smile in the "Foto" column.

Nome	Foto
Lurdes Boça	

# Aula 18 - PDO Paginação, Busca com Filtro e Information Schema

## PDO (paginação)

- O recurso/técnica de paginação em repositórios de dados permite que uma parte dos dados seja carregada por demanda do usuário.
- Com a paginação do lado do servidor, o número de registros de uma consulta também será limitada pelo limite estabelecido na paginação.
- O número e o tamanho das páginas digitais em um documento são limitados pela quantidade de dados no repositório de dados, não pelos dispositivos de vídeo ou pela quantidade de “papel”.

### aula-18/paginacao.php

```
<?php

include_once 'conexao.php';

$pagina = (isset($_REQUEST['pagina'])) ? $_REQUEST['pagina'] : 1;
$limit = 5;
$inicio = $pagina * $limit;
$offset = ($pagina - 1) * $limit;
$query = "SELECT id, titulo, preco, isbn, edicao, ano_publicacao FROM livro ";
$query_total = $pdo->query("SELECT COUNT(*) FROM ($query) q");
$total = $query_total->fetchColumn();
$statement = limit($pdo, $query, $limit, $offset);
$livros = $statement->execute();

function limit($pdo, string $query, int $limit, int $offset, string $order = 'id') :
\PDOStatement{
    $query_limit = "";

    switch ($pdo->getAttribute(PDO::ATTR_DRIVER_NAME)) {
        case 'sqlsrv': //>= 2012
        case 'oci': //>= 12c
            $query_limit = "$query ORDER BY $order OFFSET :offset ROWS FETCH
NEXT :limit ROWS ONLY";
            break;
        default: //mysql e postgres
            $query_limit = "$query ORDER BY $order LIMIT :limit OFFSET :offset";
            break;
    }
}
```

```

$statement = $pdo->prepare($query_limit);
$statement->bindValue(':offset', (int) $offset, PDO::PARAM_INT);
$statement->bindValue(':limit', (int) $limit, PDO::PARAM_INT);
return $statement;
}

function montaLinha(array $row, $tag = 'td'){
    return "<tr>".implode("",array_map(function($row) use ($tag){
        return "<$tag>" . $row . "</{$tag}>";
    }, $row))."</tr>";
}

echo "<table border>";
echo montaLinha(['ID','Título','Preço','ISBN','Edição','Ano'], 'th');

while ($row = $statement->fetch()){
    echo montaLinha($row);
}

echo "</table>";
echo (($pagina-1) > 0) ? "<a href='index.php?pagina=".($pagina-1)."'">Anterior</a>" : "Anterior";
echo "&nbsp;";
echo (($pagina)*$limit < $total) ? "<a href='index.php?pagina=".($pagina+1)."'">Próximo</a>" : "Próximo";

```

The screenshot shows a web browser window with the URL [localhost/curso-php/aula-18/paginacao.php](http://localhost/curso-php/aula-18/paginacao.php). The page displays a table of books with columns: ID, Título, Preço, ISBN, Edição, and Ano. Below the table are navigation links: 'Anterior' and 'Próximo'. The 'Próximo' link is highlighted with a purple border.

ID	Título	Preço	ISBN	Edição	Ano
1	Sucesso na Vida	39.99	123456789112	15	2012
2	Brincadeira em Excesso Virou Bobeira	44.01	9876543211112	2	2015
3	Fazendo Bolos com CakePHP	89.95	3216547894561	3	2017
4	Vamos Investigar?	63.22	7849516236295	2	2014
5	Portabilidade Manual: Um Tutorial Prático	100.99	4568521597534	2	1997

Anterior [Próximo](#)

## PDO - Paginação com Filtro

- Muitas vezes é necessário permitir que o usuário possa recuperar um subconjunto específico dos dados persistidos pelo sistema
- Com o recurso do formulário é possível permitir que o usuário possa escolher e/ou digitar determinados valores que, aplicados à query, afetará o resultado final exibido na lista/tabela
- Utilizando paginação síncrona, faz-se necessário persistir o filtro nos links da paginação permitindo que, via get, os valores possam ser reutilizados na próxima página pela query.

Neste formulário, usaremos o PHP para recuperar os valores dos campos de pesquisa, caso os mesmos tenham sido usados (e repassados pelos links da paginação e/ou submissão do próprio formulário).

### **aula-18/paginacao\_filtro.php**

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Paginação com filtro</title>
    </head>
    <body>
        <form>
            <label>Nome:</label>
            <input name="nome" value=<?php echo isset($_REQUEST['nome']) ? $_REQUEST['nome'] : "?>" />
            <label>CPF:</label>
            <input name="cpf" value=<?php echo isset($_REQUEST['cpf']) ? $_REQUEST['cpf'] : "?>" />
            <input type="submit" value="filtrar" />
        </form>

        <?php
            // put your code here
            // include 'conexao_oracle.php';
            include 'conexao.php';

            function limit(\PDO $pdo, string $query, int $limit, int $offset, string $order = 'id'): \PDOStatement {
                $query_limit = "";
                switch ($pdo->getAttribute(PDO::ATTR_DRIVER_NAME)) {
                    case 'sqlsrv':
                    case 'oci':
                        $query_limit = "$query ORDER BY $order OFFSET :offset ROWS FETCH NEXT :limit ROWS ONLY";
                }
            }
        <?php
```

```

        break;
    default:
        $query_limit = "$query ORDER BY $order LIMIT :limit OFFSET
:offset";
        break;
    }

    $statement = $pdo->prepare($query_limit);
    $statement->bindValue('offset', (int) $offset, PDO::PARAM_INT);
    $statement->bindValue('limit', (int) $limit, PDO::PARAM_INT);
    return $statement;
}

$pagina = (isset($_REQUEST['pagina'])) ? $_REQUEST['pagina'] : 1;
unset($_REQUEST['pagina']);
$parametros = [];
$nome = "";
$cpf = "";
$limit = 5;
$inicio = $pagina * $limit;
$offset = ($pagina - 1) * $limit;
$query = "SELECT * FROM funcionario WHERE 1=1 ";

if(!empty($_REQUEST['nome'])){
    $nome = $_REQUEST['nome'];
    $parametros['nome'] = $nome;
    $query .= " AND nome LIKE :nome";
}

if(!empty($_REQUEST['cpf'])){
    $cpf = $_REQUEST['cpf'];
    $parametros['nome'] = $cpf;
    $query .= " AND cpf LIKE :cpf";
}

$query_total = $pdo->prepare("SELECT COUNT(*) FROM ($query) q");
$query_total->execute($parametros);
$total = $query_total->fetchColumn();
$statement = limit($pdo, $query, $limit, $offset);
(!$nome) ?: $statement->bindValue('nome', "%$nome%");
(!$cpf) ?: $statement->bindValue('cpf', "%$cpf%");
$funcionarios = $statement->execute();

echo "<table border>";
echo "<tr><th>Nome</th><th>CPF</th></tr>";
while($funcionario = $statement->fetch()){
    echo "<tr>";
    echo "<td>{$funcionario['nome']}</td>";
    echo "<td>{$funcionario['cpf']}</td>";
    echo "</tr>";
}
echo "</table>";

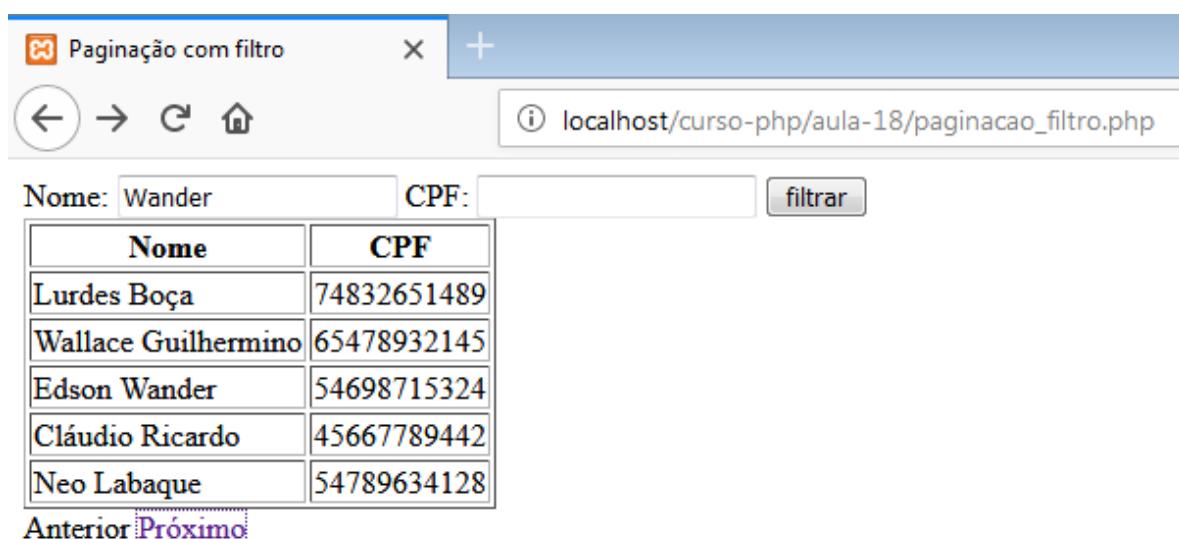
```

```

$parametrosUrl = http_build_query($_REQUEST);
echo (( $pagina - 1 ) > 0 ) ? "<a href='paginacao_1.php?pagina=". ($pagina-1) . "&$parametrosUrl'>Anterior</a>" : "Anterior";
echo "&nbsp";
echo (( $pagina ) * $limit < $total) ? "<a href='paginacao_1.php?pagina=". ($pagina+1) . "&$parametrosUrl'>Próximo</a>" :
"Próximo";
?>

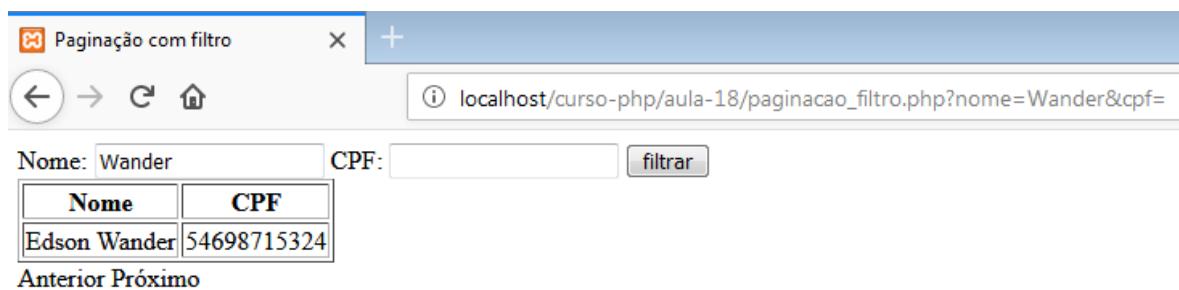
</body>
</html>

```



Nome	CPF
Lurdes Boça	74832651489
Wallace Guilhermino	65478932145
Edson Wander	54698715324
Cláudio Ricardo	45667789442
Neo Labaque	54789634128

Anterior [Próximo](#)



Nome	CPF
Edson Wander	54698715324

Anterior [Próximo](#)

## PDO e Information Schema

- Com o Information Schema é possível fazer uma introspecção nas tabelas do SGBD.
- Informações de metadados como campos obrigatórios, tipos e ou até limites de tamanho podem ser obtidos para, por exemplo, validar entrada dos usuários de forma dinâmica.
- Tal recurso é primordial para tornar dinâmico algumas tarefas repetitivas como validação de campos e essencial na construção de uma biblioteca ORM.

Neste exemplo será recuperado os metadados de cada coluna da tabela editora. Como o Oracle não suporta o Information Schema ANSI, faz-se necessário uma adaptação para utilizar seu equivalente.

Para compatibilizar a caixa utilizada por alguns SGBDs, o array\_change\_key\_case com a constante CASE\_UPPER manterá todos os índices do resultado do fetchAll() em caixa alta (padrão ansi do information schema)

### **aula-18/information\_schema.php**

```
<?php

// include 'conexao_oracle.php';
include 'conexao.php';

$driver = $pdo->getAttribute(\PDO::ATTR_DRIVER_NAME);
$table = 'editora';
$colunas = 'COLUMN_NAME, IS_NULLABLE, DATA_TYPE,
CHARACTER_MAXIMUM_LENGTH';
$from = 'information_schema.COLUMNS';

if($driver == 'oci'){
    $table = strtoupper($table);
    $from = 'all_tab_columns';
    $colunas = 'column_name, nullable as IS_NULLABLE, data_type, data_length as
CHARACTER_MAXIMUM_LENGTH';
}

$query = "SELECT $colunas " . " FROM $from WHERE table_name = ?";
$statement = $pdo->prepare($query);
$statement->execute([$table]);
$schema = $statement->fetchAll();
```

```

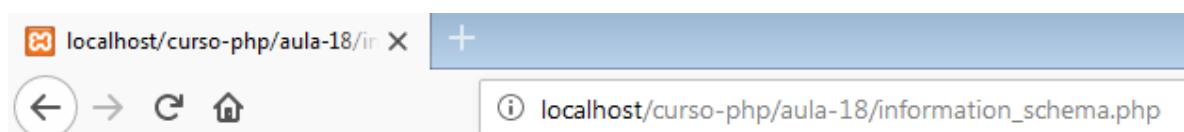
echo "<table>";
echo "<tr><th>name</th><th>is
null?</th><th>type</th><th>size</th></tr>";

foreach($schema as $coluna_schema){
    $coluna_schema = array_change_key_case($coluna_schema, CASE_UPPER);

    echo "<tr>";
    echo "<td>{$coluna_schema['COLUMN_NAME']}</td>";
    echo "<td>{$coluna_schema['IS_NULLABLE']}</td>";
    echo "<td>{$coluna_schema['DATA_TYPE']}</td>";
    echo "<td>{$coluna_schema['CHARACTER_MAXIMUM_LENGTH']}</td>";
    echo "</tr>";
}

echo "</table>";

```



<b>name</b>	<b>is null?</b>	<b>type</b>	<b>size</b>
id	NO	int	
nome	NO	varchar	255
website	NO	varchar	255
cnpj	NO	varchar	14
endereco	NO	varchar	255

# Aula 19 - Estruturas de Dados com SPL Classes Pilha, Fila, Lista, Árvore e Mapa

Estruturas de Dados com Classes SPL (Standard PHP Library)

- A Biblioteca Padrão do PHP (SPL) é uma coleção de interfaces e classes que se destinam a solucionar problemas comuns.
- Nenhuma biblioteca externa é necessária para construir essa extensão e está disponível e compilada por padrão desde o PHP 5.0.
- A SPL fornece um conjunto de:
  - Estrutura de dados (array, pilha, fila etc.)
  - Iteradores para percorrer objetos,
  - Interfaces,
  - Exceptions

## SplFixedArray

- A classe SplFixedArray fornece as principais funcionalidades do array. A principal diferença entre um SplFixedArray e um array PHP normal é que o SplFixedArray é de tamanho fixo e permite apenas inteiros como valores para os índices.
- Notoriamente o SplFixedArray consome um número menor de memória RAM e, muitas vezes, também é mais rápido quando comparado ao array “tradicional” do PHP.

## **aula-19/aula19a.php**

```
<?php

$fixed = new SplFixedArray(3);
$fixed[0] = 'laranja';
$fixed[1] = 'maçã';
$fixed[2] = 'banana';

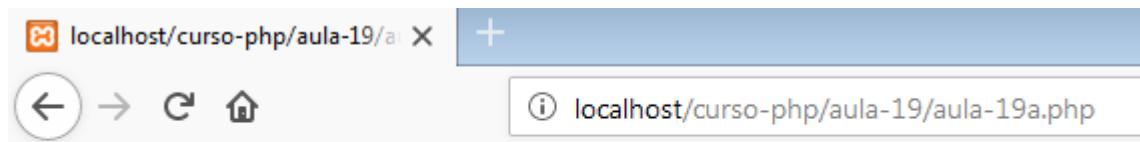
foreach ($fixed as $item) {
    echo $item . "<br>";
}

echo "<hr>";

echo "<p>Diminuindo o array</p>";

$fixed->setSize(2); //diminuindo o array

foreach ($fixed as $item) {
    echo $item . "<br>";
}
```

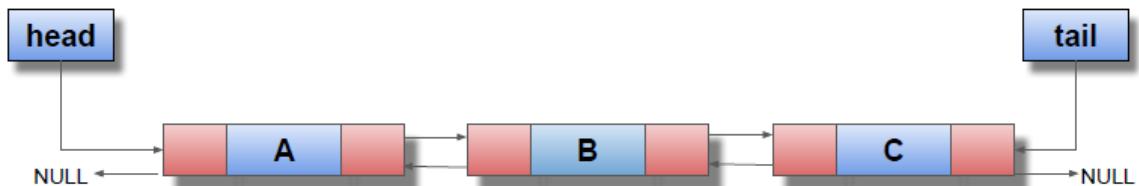


Diminuindo o array

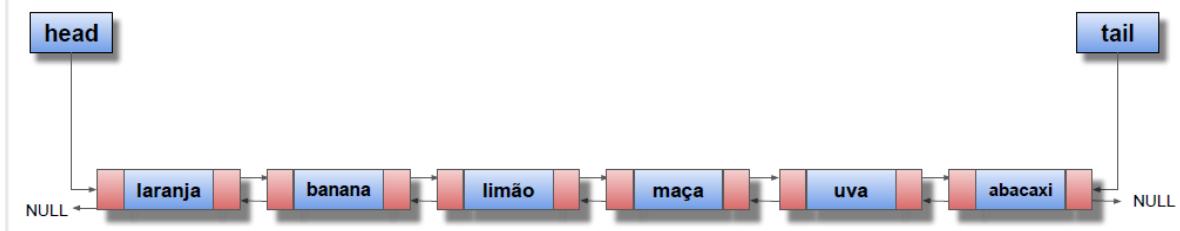
laranja  
maçã

## **SplDoublyLinkedList (Lista Duplamente Encadeada)**

- Classe genérica que representa a estrutura de uma fila duplamente encadeada.
- Permite a iteração bidirecional da lista e implementa as mesmas interfaces que o SplFixedArray.
- Ela também serve como a classe base para as classes SplStack e SplQueue, que também implementam as estruturas de dados de pilha e fila.
- Através do método setIteratorMode, o Lista encadeada pode ser organizada usando FIFO ou LIFO.
- Esta implementação não é circular, por isso, não existe uma referência entre os elementos das extremidades.



## SplDoublyLinkedList (iteração, cabeça e cauda)



### aula-19/aula19b.php

```
<?php

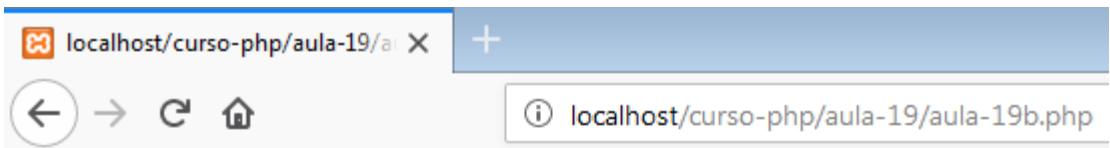
$dll = new \SplDoublyLinkedList();

$dll->push("laranja");
$dll->push("banana");
$dll->push("limão");
$dll->push("maçã");
$dll->push("uva");
$dll->push("abacaxi");

echo "Cabeça: ". $dll->bottom(). "<br/>";
echo "Cauda: ". $dll->top(). "<br/>";

$prev = null;
$dll->rewind(); //rebobinando

while ($dll->valid()) {
    $current = $dll->current();
    echo 'Anterior: '.$prev, "<br/>";
    echo 'Atual: '.$current, "<br/>";
    $prev = $current;
    $dll->next();
    $next = $dll->current();
    echo 'Próximo: '.$next. "<br/>";
    echo "<br/>";
}
```



Cabeça: laranja

Cauda: abacaxi

Anterior:

Atual: laranja

Próximo: banana

Anterior: laranja

Atual: banana

Próximo: limão

Anterior: banana

Atual: limão

Próximo: maçã

Anterior: limão

Atual: maçã

Próximo: uva

Anterior: maçã

Atual: uva

Próximo: abacaxi

Anterior: uva

Atual: abacaxi

Próximo:

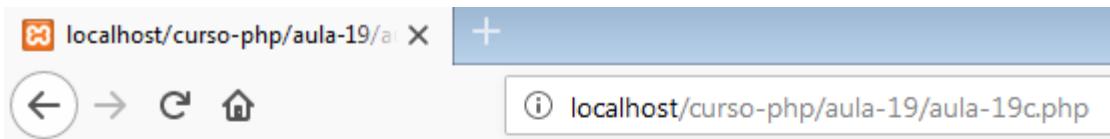
## **SplDoublyLinkedList (métodos)**

### **aula-19/aula19c.php**

```
<?php

$dll = new \SplDoublyLinkedList();
$dll->unshift(200); //no início
imprimir($dll);
$dll->unshift(100); //no início
imprimir($dll);
$dll->push(34); //no fim
imprimir($dll);
$dll->push(35); //no fim
imprimir($dll);
$dll->add(2, 3); //posição específica
imprimir($dll);
$dll->unshift(670); //no início
imprimir($dll);
$dll->add(3, 450); //posição específica
imprimir($dll);
$dll->pop(); //remove o último
imprimir($dll);
$dll->shift(); //remove o primeiro
imprimir($dll);
$dll->offsetUnset(1); //remove de posição específica (2a)
imprimir($dll);

function imprimir(\SplDoublyLinkedList &$dll) {
    $dll->rewind();
    $values = [];
    while ($dll->valid()) {
        $values[] = $dll->current();
        $dll->next();
    }
    echo "[ " . implode(' , ', $values) . " ] </br>";
}
```



```
[200]  
[100, 200]  
[100, 200, 34]  
[100, 200, 34, 35]  
[100, 200, 3, 34, 35]  
[670, 100, 200, 3, 34, 35]  
[670, 100, 200, 450, 3, 34, 35]  
[670, 100, 200, 450, 3, 34]  
[100, 200, 450, 3, 34]  
[100, 450, 3, 34]
```

## **FIFO e LIFO**

- FIFO: Comportamento First in, first out (primeiro a entrar, primeiro a sair) [Fila / Queue];
- Exemplos de FIFO:
  - Scheduling de CPU e Disco.
  - Fila da impressora: os trabalhos enviados para a impressora são impressos na ordem de chegada.
  - Transferindo dados de maneira assíncrona entre dois processos (E / S de Arquivo, Pipes, buffers de E / S, etc.)
  - App de gerenciamento fila em bilheteiras, hospitais etc.
- Comportamento Last in, first out (última a entrar, primeiro a sair). [Pilha / Stack]
- Exemplos de LIFO:
  - Mecanismo Desfazer / Refazer em editores de texto e outras aplicações (CTRL+Z);
  - Verificação de sintaxe do compilador para chaves correspondentes
  - Uma pilha de pratos ou livros ou cadeiras.
  - Backtracking: Este é um processo quando você precisa acessar o elemento de dados mais recente em uma série de elementos.
  - Inverter uma palavra

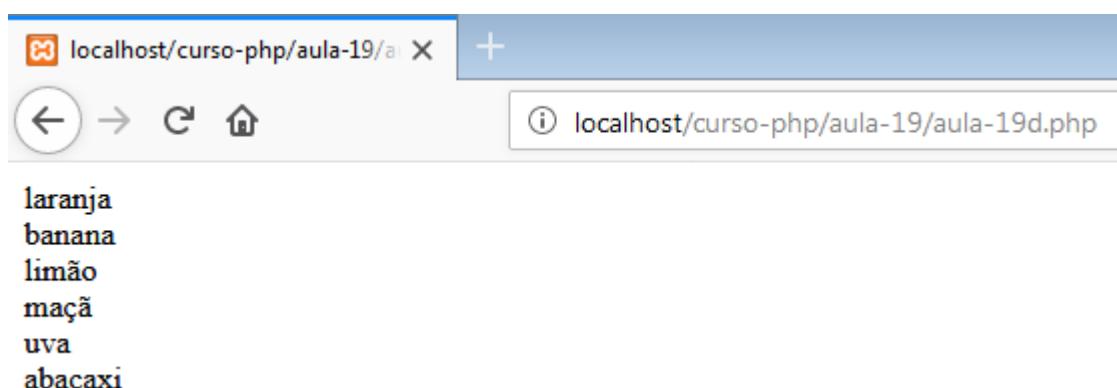
## SplDoublyLinkedList (modo FIFO)

### aula-19/aula19d.php

```
<?php

//FIFO
$fifo = new \SplDoublyLinkedList();
$fifo->setIteratorMode(\SplDoublyLinkedList::IT_MODE_FIFO);
$fifo->push("laranja");
$fifo->push("banana");
$fifo->push("limão");
$fifo->push("maçã");
$fifo->push("uva");
$fifo->push("abacaxi");

foreach ($fifo as $value) {
    echo $value."<br>";
}
```



## **SplDoublyLinkedList (Modo LIFO)**

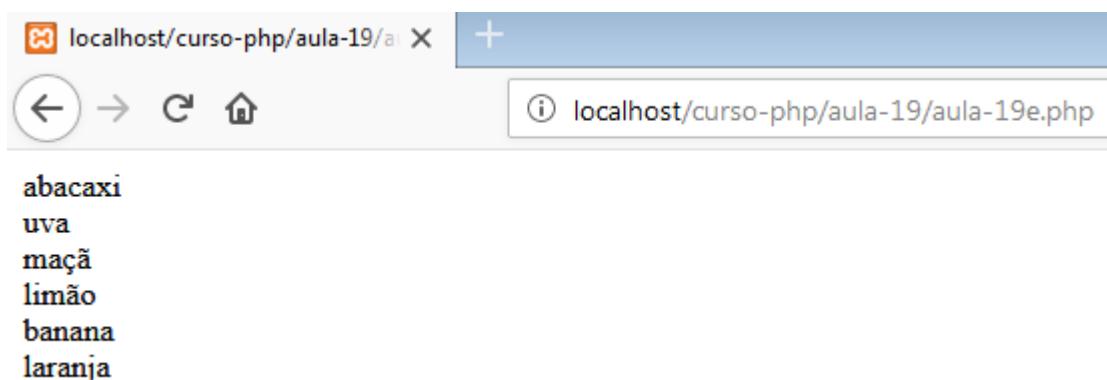
### **aula-19/aula19e.php**

```
<?php

//FIFO
$fifo = new \SplDoublyLinkedList();
$fifo->setIteratorMode(\SplDoublyLinkedList::IT_MODE_LIFO);

$fifo->push("laranja");
$fifo->push("banana");
$fifo->push("limão");
$fifo->push("maçã");
$fifo->push("uva");
$fifo->push("abacaxi");

foreach ($fifo as $value) {
    echo $value."<br/>";
}
```



## **SplDoublyLinkedList (Modo Delete)**

Com o modo de Iteração Delete, os item são removidos da lista duplamente encadeadas, conforme os mesmos são iterados. O modo default é KEEP, que mantém os itens na lista.

## **aula-19/aula19f.php**

```
<?php

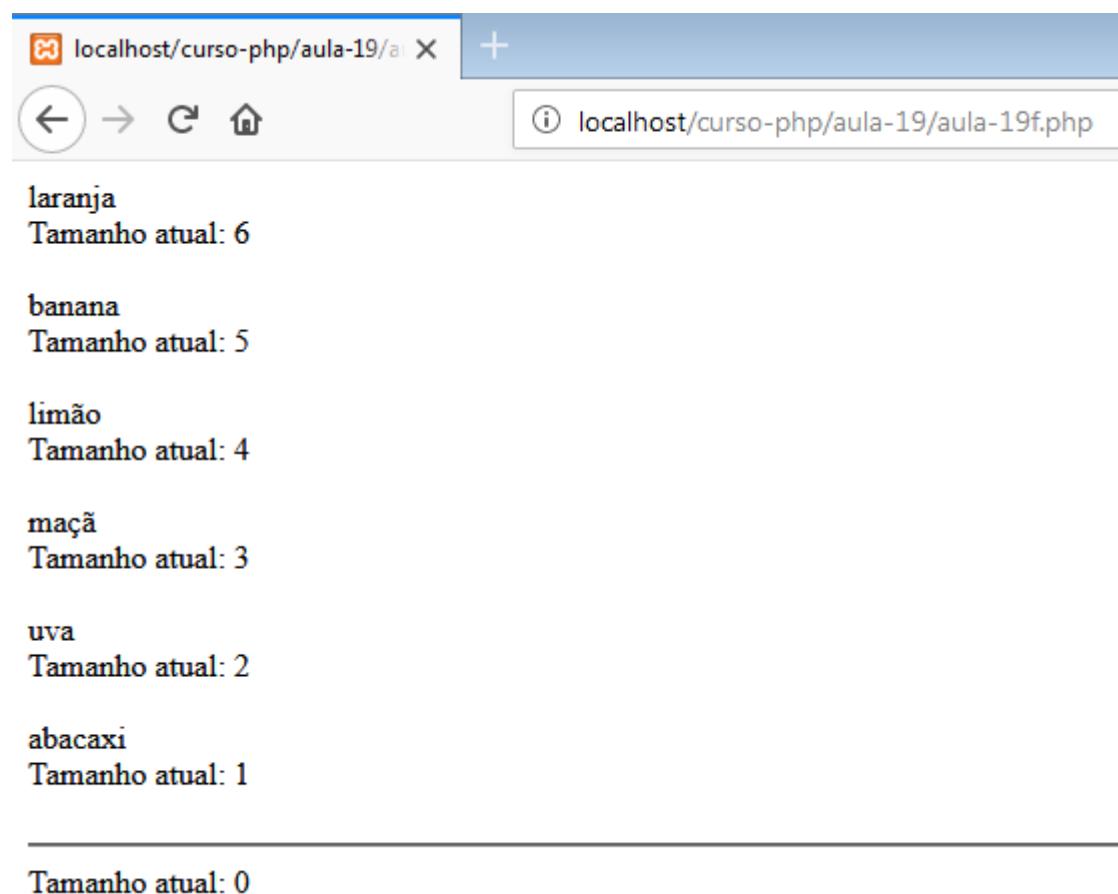
$dll = new \SplDoublyLinkedList();
$dll->setIteratorMode(\SplDoublyLinkedList::IT_MODE_DELETE);

$dll->push("laranja");
$dll->push("banana");
$dll->push("limão");
$dll->push("maçã");
$dll->push("uva");
$dll->push("abacaxi");

foreach ($dll as $value) {
    echo $value."<br>";
    echo "Tamanho atual: ".$dll->count()."<br><br>";
}

echo "<hr>";

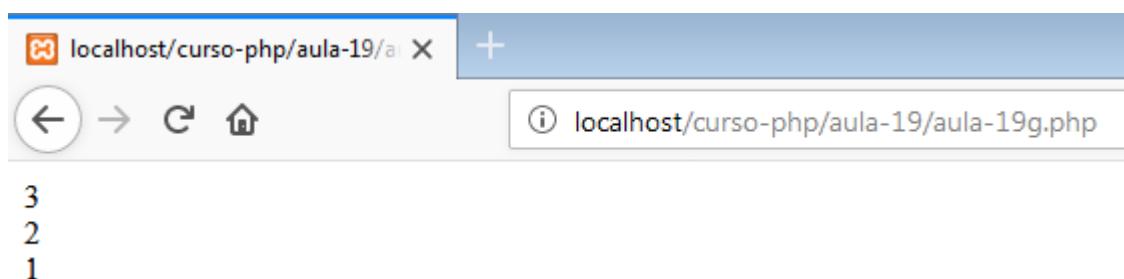
echo "Tamanho atual: ".$dll->count();
```



## Pilha (SplStack) / LIFO

### aula-19/aula19g.php

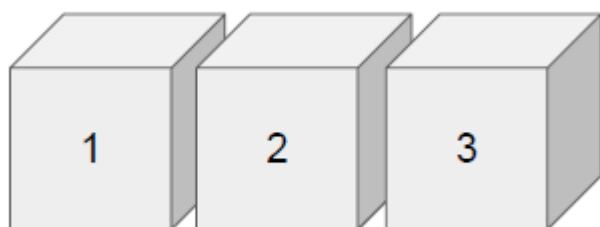
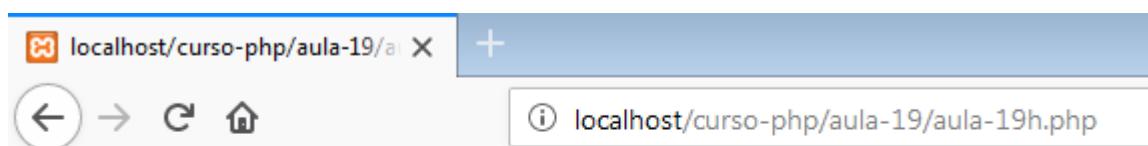
```
<?php  
  
$stack = new SplStack();  
  
$stack[] = 1;  
$stack[] = 2;  
$stack[] = 3;  
  
foreach ($stack as $item) {  
    echo $item . "<br>";  
}
```



## Fila (SplQueue) / FIFO

### aula-19/aula19h.php

```
<?php  
  
$queue = new SplQueue();  
  
$queue[] = 1;  
$queue[] = 2;  
$queue[] = 3;  
  
foreach ($queue as $item) {  
    echo $item . "<br>";  
}
```



## **SplHeap**

- Heap é uma estrutura de dados baseada em árvore especializada que é essencialmente uma árvore quase completa que satisfaz a propriedade heap: em um heap máximo, para qualquer nó dado C, se P é um nó pai de C, então a chave (o valor) de P é maior que ou igual à chave de C. Em uma pilha mínima (MinHeap), a chave de P é menor ou igual à chave de C.
- O nó no "topo" do heap (sem pais) é chamado de nó raiz.
- O heap é uma implementação maximamente eficiente de um tipo de dados abstrato chamado de fila de prioridade e, na verdade, as filas de prioridade são geralmente chamadas de "heaps", independentemente de como elas podem ser implementadas. Em um heap, o elemento de prioridade mais alto (ou mais baixo) é sempre armazenado na raiz.
- No entanto, um heap não é uma estrutura classificada; pode ser considerado parcialmente ordenado.
- Um heap é uma estrutura de dados útil quando é necessário remover repetidamente o objeto/item com a prioridade mais alta (ou mais baixa).

## **Spl Heap (Exemplo)**

## **aula-19/aula19i.php**

```
<?php

class CampeonatoBrasileiro extends SplHeap {
    protected function compare($value1, $value2): int {
        if ($value1['pontuacao'] == $value2["pontuacao"]) {
            return $value1['vitorias'] <=> $value2['vitorias'];
        }
        return $value1['pontuacao'] <=> $value2['pontuacao'];
    }
}

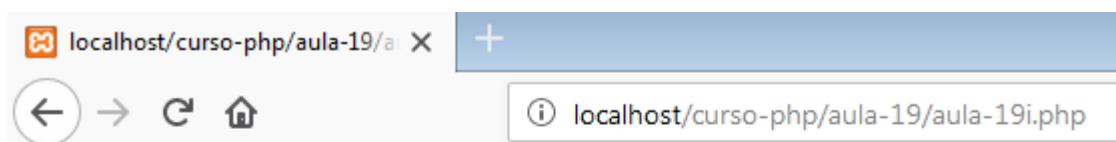
$heap = new CampeonatoBrasileiro();

$heap->insert(['nome' => 'Remo', 'pontuacao' => 22, 'vitorias' => 6]);
$heap->insert(['nome' => 'Santa Cruz', 'pontuacao' => 28, 'vitorias' => 7]);
$heap->insert(['nome' => 'Atlético AC', 'pontuacao' => 30, 'vitorias' => 9]);
$heap->insert(['nome' => 'Botafogo PB', 'pontuacao' => 26, 'vitorias' => 6]);
$heap->insert(['nome' => 'Náutico', 'pontuacao' => 31, 'vitorias' => 9]);
$heap->insert(['nome' => 'Confiança', 'pontuacao' => 23, 'vitorias' => 5]);
$heap->insert(['nome' => 'Globo', 'pontuacao' => 22, 'vitorias' => 4]);

//Campeão:
echo "Campeão da Série C 2018: {$heap->top()['nome']} <br/><br/>";

//Resultados
$total = $heap->count();

foreach ($heap as $key => $time) {
    echo ($total - $key) . "º {$time['nome']} <br/>";
}
```

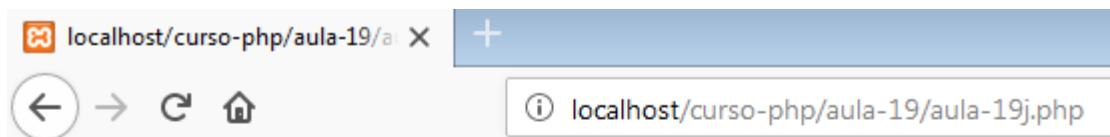


- 1º Náutico
- 2º Atlético AC
- 3º Santa Cruz
- 4º Botafogo PB
- 5º Confiança
- 6º Remo
- 7º Globo

## MinHeap (SplMinHeap)

### aula-19/aula19j.php

```
<?php  
  
$minHeap = new \SplMinHeap();  
  
$minHeap->insert(30);  
$minHeap->insert(50);  
$minHeap->insert(10);  
$minHeap->insert(105);  
$minHeap->insert(99);  
$minHeap->insert(88);  
  
echo "mínimo:". $minHeap->top();  
echo "<br/><br/>";  
  
foreach ($minHeap as $value) {  
    echo $value."<br/>";  
}  
?
```



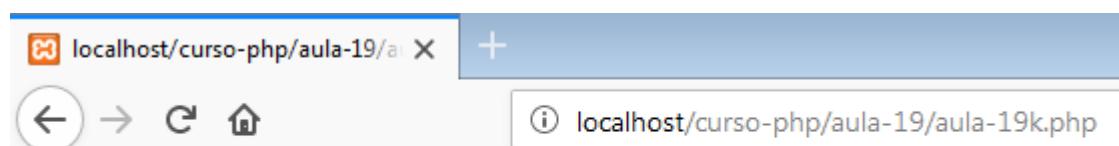
mínimo:10

10  
30  
50  
88  
99  
105

## MaxHeap (SplMaxHeap)

### aula-19/aula19k.php

```
<?php  
  
$maxHeap = new \SplMaxHeap();  
  
$maxHeap->insert(30);  
$maxHeap->insert(50);  
$maxHeap->insert(10);  
$maxHeap->insert(105);  
$maxHeap->insert(99);  
$maxHeap->insert(88);  
  
echo "max:". $maxHeap->top();  
echo "<br/><br/>";  
  
foreach ($maxHeap as $value) {  
    echo $value."<br/>";  
}  
?
```

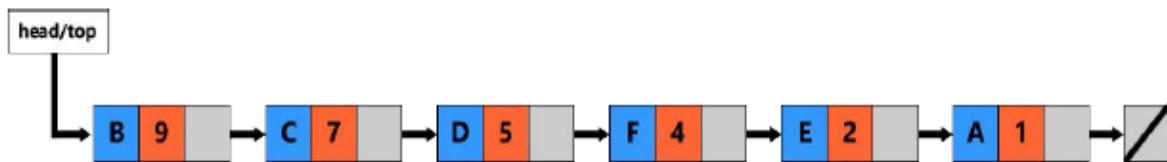


max:105

105  
99  
88  
50  
30  
10

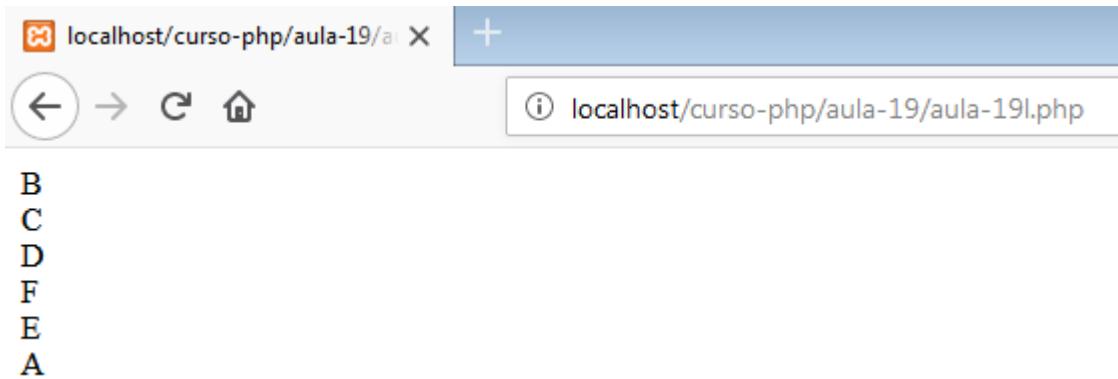
## SplPriorityQueue (Fila de Prioridade)

- Fila de prioridade é uma extensão da fila com as seguintes características:
  - Cada item tem uma prioridade associada a ele.
  - Um elemento com alta prioridade é retirado da fila antes de um elemento com baixa prioridade.
  - Se dois elementos tiverem a mesma prioridade, eles serão atendidos de acordo com sua ordem na fila.
- Filas de prioridade são estruturas de dados úteis em simulações, particularmente para manter um conjunto de eventos futuros ordenados pelo tempo, para que possamos recuperar rapidamente o que acontecerá. Eles são chamados filas de prioridade porque permitem recuperar itens não pelo tempo de inserção (como em uma pilha ou fila), nem por uma correspondência de chave (como em um dicionário), mas por qual item tem a prioridade mais alta de recuperação.



### aula-19/aula19I.php

```
<?php  
  
$prQueue = new SplPriorityQueue();  
  
$prQueue->insert('A',1);  
$prQueue->insert('B',9);  
$prQueue->insert('C',7);  
$prQueue->insert('D',5);  
$prQueue->insert('E',2);  
$prQueue->insert('F',4);  
  
foreach ($prQueue as $item) {  
    echo $item . "<br/>";  
}  
?
```



## SplPriorityQueue (Flags)

Com o uso do método `setExtractFlags` é possível determinar qual será o resultado extraído ao recuperar os dados de um `SplPriorityQueue`. Seja com os dados (default), apenas o valor da priorização (priority) ou ambos (both).

### aula-19/aula19m.php

```
<?php

//flags

$prQueue = new SplPriorityQueue();

$prQueue->insert('A',1);
$prQueue->insert('B',9);
$prQueue->insert('C',7);
$prQueue->insert('D',5);
$prQueue->insert('E',2);
$prQueue->insert('F',4);
$prQueue->insert('G',5);

$prQueue->setExtractFlags(SplPriorityQueue::EXTR_PRIORITY);

foreach ($prQueue as $item) {
    echo $item."<br/>";
}
```

```
localhost/curso-php/aula-19/aula-19m.php
9
7
5
5
4
2
1
```

## aula-19/aula19n.php

```
<?php

//flags
$prQueue = new SplPriorityQueue();

$prQueue->insert('A',1);
$prQueue->insert('B',9);
$prQueue->insert('C',7);
$prQueue->insert('D',5);
$prQueue->insert('E',2);
$prQueue->insert('F',4);
$prQueue->insert('G',5);

$prQueue->setExtractFlags(SplPriorityQueue::EXTR_BOTH);

foreach ($prQueue as $item) {
    echo "prioridade: {$item['priority']} ; dado:{$item['data']}<br/>";
}
```

```
localhost/curso-php/aula-19/aula-19n.php
prioridade: 9 ; dado:B
prioridade: 7 ; dado:C
prioridade: 5 ; dado:D
prioridade: 5 ; dado:G
prioridade: 4 ; dado:F
prioridade: 2 ; dado:E
prioridade: 1 ; dado:A
```

## **SplStorageObject**

- A classe SplObjectStorage fornece um mapa de objetos para dados ou, ignorando dados, um conjunto de objetos. Esse duplo propósito pode ser útil em muitos casos, envolvendo a necessidade de identificar objetos de forma exclusiva.
- Diferente de um hashmap, o SplObjectStorage não atua como um armazenamento de chave-valor, mas apenas um conjunto de objetos. Algo está no set ou não, mas sua posição não é importante.
- A "chave" de um elemento no SplObjectStorage é, na verdade, o hash do objeto. Isso faz com que não seja possível adicionar várias cópias da mesma instância de objeto a um SplObjectStorage, para que você não precise verificar se já existe uma cópia antes de adicionar.
- A principal vantagem do SplObjectStorage é o fato de que você ganha muitos métodos para lidar e interagir com diferentes conjuntos (contains(), removeAll(), removeAllExcept () etc).

### **aula-19/aula19o.php**

```
<?php
class Pessoa {
    public $nome;
}

$storage = new SplObjectStorage();

$o1 = new Pessoa();
$o2 = new Pessoa();
$o3 = new Pessoa();
$o4 = new Pessoa();

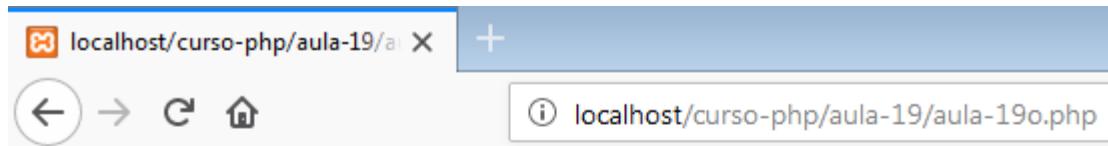
$o1->nome = 'João';
$o2->nome = 'Maria';
$o3->nome = 'Thiago';
$o4->nome = 'Maria';

$storage->attach($o1);
$storage->attach($o2);
$storage->attach($o3);
$storage->attach($o1); //já existe, mesmo hash
$storage->attach($o4); //estados iguais, objetos diferentes

$storage->detach($o3); //removendo
//var_dump($storage[0]); //nulo !

echo "contém o1? ".var_export($storage->contains($o1), true)."</br>";
```

```
foreach($storage as $key => $o){  
    echo "<pre>";  
    print_r($o); //aqui funciona!  
    echo "</pre>";  
}
```



```
contém o1? true  
  
Pessoa Object  
(  
    [nome] => João  
)  
  
Pessoa Object  
(  
    [nome] => Maria  
)  
  
Pessoa Object  
(  
    [nome] => Maria  
)
```

## **SPLEnum (Conjunto finito de identificadores)**

Enum é um tipo de dado definido pelo usuário que consiste em um conjunto de constantes nomeadas chamadas enumeradores.

O SPLenum faz parte de uma extensão PECL chamada SPLTypes. Infelizmente esta extensão não é mais compatível com PHP 7 >=, por isso, faz-se necessário utilizar algum fork (e compilar) ou um polyfill como por exemplo o duck-projects.

### **aula-19/aula19p.php**

```
<?php

class MesesDoAno extends SplEnum {
    const __default = self::Janeiro;
    const Janeiro = 1; const Fevereiro = 2;
    const Marco = 3; const Abril = 4;
    const Maio = 5; const Junho = 6;
    const Julho = 7; const Agosto = 8;
    const Setembro = 9; const Outubro = 10;
    const Novembro = 11; const Dezembro = 12;
}

echo new MesesDoAno(MesesDoAno::Junho) . "<br/>";
echo MesesDoAno::Novembro . "<br/>";

echo "<pre>";
print_r((new MesesDoAno())->getConstList(true));
echo "</pre>";

try {
    new MesesDoAno(13);
} catch (UnexpectedValueException $uve) {
    echo $uve->getMessage() . "<br>";
}
```