



# Desenvolvimento de Aplicações WEB

## *PHP*

Profa. Joyce Miranda

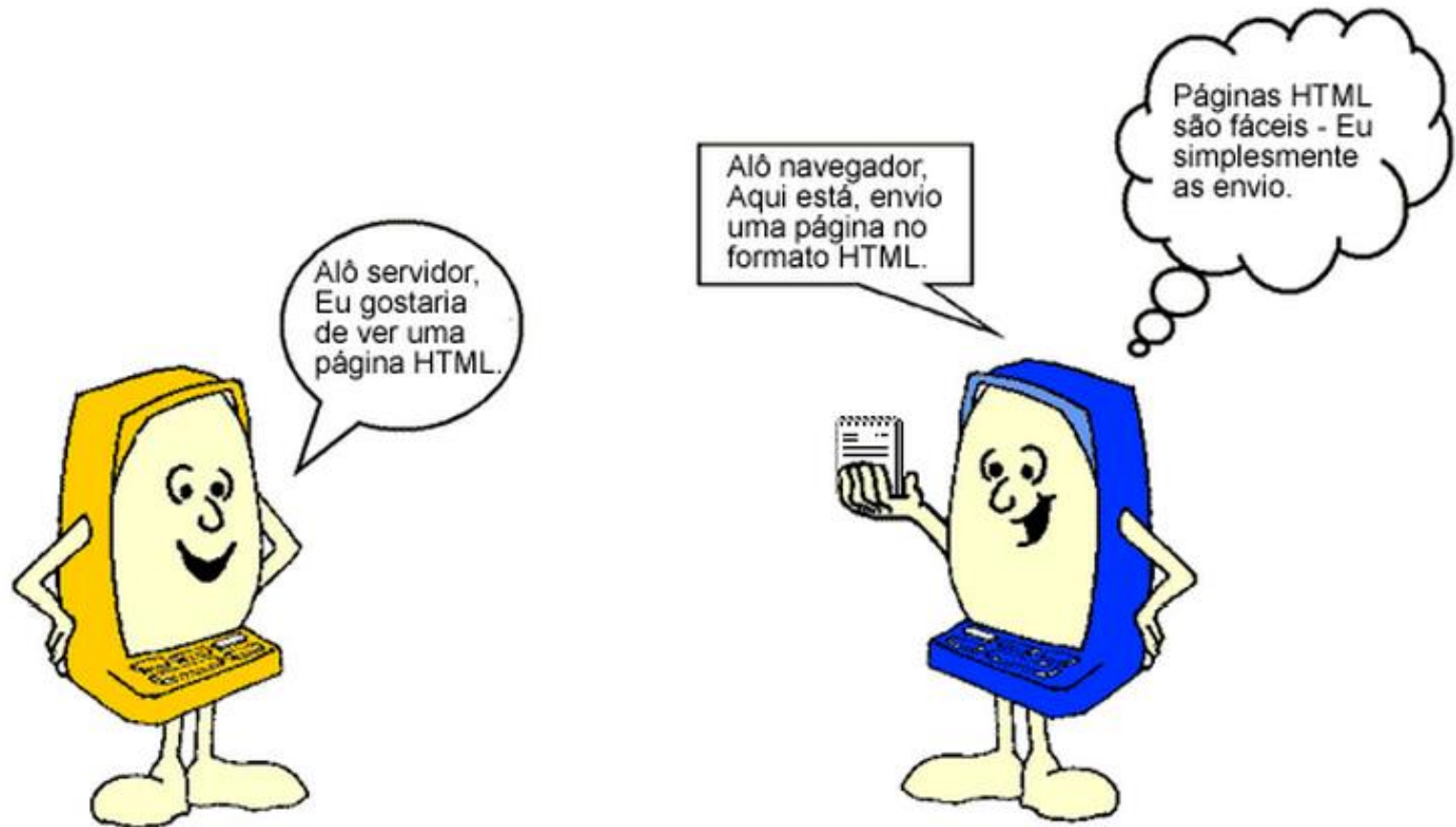
## PHP - *PHP: Hypertext Preprocessor*

---

- ▶ É uma linguagem back-end
- ▶ É utilizada para apresentar conteúdo nas páginas web.
- ▶ Algumas características
  - ▶ Multiplataforma
  - ▶ *Open Source*
  - ▶ *Script server-side*
  - ▶ Interpretada e não compilada
  - ▶ Permite programação estruturada e orientada a objetos

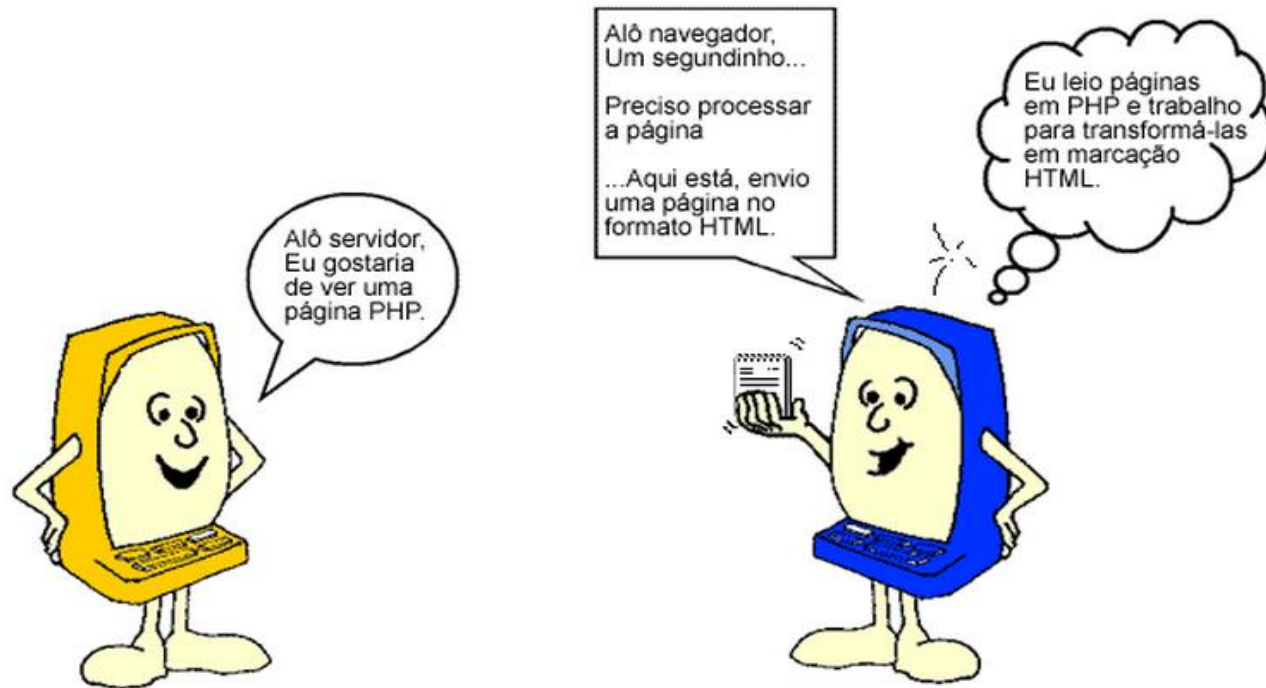
## ► Como funciona?

- Requisitando uma página HTML : <http://www.meusite.com/page.html>



► Como funciona?

- Requisitando uma página PHP: <http://www.meusite.com/page.php>



- O servidor executa as tarefas e retorna um resultado HTML
- Assim, não é possível visualizar os scripts PHP inseridos em uma página

- ▶ Do que você precisa para começar?
  - ▶ Navegador
  - ▶ Servidor WEB (Ex: Apache)
  - ▶ PHP
  - ▶ Banco de Dados (Ex: MySQL)
  - ▶ Pacotes pré-configurados
    - ▶ **WAMP** - <http://www.wampserver.com>
    - ▶ **EasyPHP DevServer** - <http://www.easyphp.org>
    - ▶ **XAMPP** - <http://www.apachefriends.org>

## ▶ Hello PHP!!

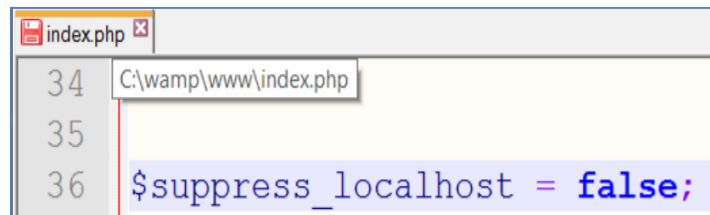
### ▶ Conteúdo de uma página PHP

- ▶ Tags HTML
- ▶ Conteúdo
- ▶ Código PHP

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
      echo "HELLO PHP !!";
    ?>
  </body>
</html>
```

## ▶ Hello PHP!!

- ▶ Inicie o servidor WEB
- ▶ Crie uma pasta para seu site dentro do servidor. Ex: [meuProjetoPHP](#)
- ▶ Crie seu código e salve o arquivo com a extensão “.php” dentro da pasta criada. Ex: [hello.php](#)
- ▶ Acesse pelo browser o arquivo pelo endereço de seu servidor web
  - ▶ <http://localhost/meuProjetoPHP/hello.php>
  - ▶ <http://127.0.0.1/meuProjetoPHP/hello.php>



```
index.php
34 C:\wamp\www\index.php
35
36 $suppress_localhost = false;
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
      echo "HELLO PHP !!";
    ?>
  </body>
</html>
```

## ► Sintaxe

### ► Comentários

```
<?php
    echo 'Isto é um teste'; // Estilo de comentário de uma linha em  c++
    /* Este é um comentário de múltiplas linhas
       ainda outra linha de comentário */
    echo 'Isto é ainda outro teste';
    echo 'Um teste final'; # Este é um comentário de uma linha no estilo shell
?>
```



## ► Sintaxe

### ► Variáveis

- Deve ser antecedita por um \$
- Tipagem dinâmica
- Tipos básicos:
  - boolean
  - integer
  - float (ou também double)
  - string

```
$a_bool = TRUE;    // um booleano  
$a_str  = "foo";   // uma string  
$a_str2 = 'foo';   // uma string  
$an_int = 12;      // um inteiro
```

## ► Sintaxe

### ► Operadores de Atribuição

Atribuição	O mesmo que..
$x = y$	$x = y$
$x += y$	$x = x + y$
$x -= y$	$x = x - y$
$x *= y$	$x = x * y$
$x /= y$	$x = x / y$
$x \% = y$	$x = x \% y$

## ► Sintaxe

### ► Operadores Aritméticos

Operador	Nome	Exemplo
+	Adição	$\$x + \$y$
-	Subtração	$\$x - \$y$
*	Multiplicação	$\$x * \$y$
/	Divisão	$\$x / \$y$
%	Módulo	$\$x \% \$y$
**	Exponenciação	$\$x ** \$y$

## ► Sintaxe

### ► Operadores de Incremento e Decremento

Operador	Nome
<code>++\$x</code>	Pré-incremento
<code>\$x++</code>	Pós-incremento
<code>--\$x</code>	Pré-decremento
<code>\$x--</code>	Pós-decremento

## ► Sintaxe

### ► Operadores de Comparação

Operador	Nome	Exemplo
==	Igual	\$x == \$y
===	Idêntico	\$x === \$y
!=	Diferente	\$x != \$y
<>	Diferente	\$x <> \$y
!==	Não Idêntico	\$x !== \$y
>	Maior	\$x > \$y
<	Menor	\$x < \$y
>=	Maior ou igual	\$x >= \$y
<=	Menor ou igual	\$x <= \$y

## ► Sintaxe

### ► Operadores Lógicos

Operador	Nome	Exemplo
and	And	\$x and \$y
or	Or	\$x or \$y
xor	Xor	\$x xor \$y
&&	And	\$x && \$y
	Or	\$x    \$y
!	Not	!\$x

## ► Sintaxe

### ► Conversão de Tipos

#### ► cast

(int), (integer)  
(bool), (boolean)  
(float), (double), (real)  
(string)  
(binary)  
(array)  
(object)  
(unset)

```
$y = 2.5;  
$y = (int) $y;  
var_dump($y);
```

```
int 2
```

## ► Sintaxe

### ► Operadores de String

Operador	Nome	Exemplo	Exemplo
.	Concatenação	<code>\$txt1 = "Hello"</code> <code>\$txt2 = \$txt1 . " world!"</code>	"Hello world!"
.=	Atribuição de Concatenação	<code>\$txt1 = "Hello"</code> <code>\$txt1 .= " world!"</code>	"Hello world!"



## ► Sintaxe

### ► Strings

#### ► Concatenando

```
$num = 100;  
$numDesc = "cem";  
echo "O número é $num : $numDesc ";  
//O número é 100 : cem
```

```
echo "O número é ".$num." : ".$numDesc;
```

- ▶ Sintaxe
  - ▶ Funções



## ► Sintaxe

### ► Funções

```
// Função sem parâmetros e sem retorno  
function exibe_frase () {  
    echo 'Eu sou uma função que executa a ação.';  
}  
  
// Chama a função  
exibe_frase();
```

## ► Sintaxe

### ► Funções

```
// Criei uma função para imprimir o nome de alguém,  
// então também criei o parâmetro $nome, para passar  
// o nome de alguém para dentro da função.  
function ola($nome) {  
    print 'oi ' . $nome . '!';  
}  
  
// Agora eu executo a função passando o nome de alguém para dentro dela.  
// Aqui a função imprime 'oi Eduardo!'  
ola('Eduardo');
```

## ► Sintaxe

### ► Funções

```
<?php

function retornaMaior($valor1, $valor2) {
    if( $valor1 > $valor2) {
        return $valor1;
    } else {
        return $valor2;
    }
}

echo retornaMaior(10, 5);

?>
```

## ▶ Exercício 1

- ▶ Crie uma função que vai receber como parâmetro o ano de nascimento de uma pessoa e vai retornar a idade da pessoa.

- ▶ *calculaIdade(1915)*

- ▶ *Idade: 100 anos*

- ▶ Execute a função criada.

```
//recuperar ano atual  
$anoAtual = date("Y");
```

## ► Sintaxe

- Funções Internas – Strings
- Letras Maiúsculas e Minúsculas

- **strtoupper**

- ☐ Converte as letras de um texto para letras maiúsculas.

```
$nome = 'Eduardo Monteiro';  
$nome= strtoupper($nome);  
  
// imprime EDUARDO MONTEIRO  
print $nome;
```

## ► Sintaxe

- Funções Internas - Strings
- Letras Maiúsculas e Minúsculas

### ► **strtolower**

- Converte as letras de um texto para letras minúsculas.

```
$nome = 'Eduardo Monteiro';  
$nome = strtolower($nome);  
  
// imprime eduardo monteiro  
print $nome;
```



## ► Sintaxe

### ► Funções Internas – Strings

### ► Retirar espaços

#### ► **trim**

- Retira os espaços encontrados em ambos os lados de um texto.

```
$texto = " texto com espaços ";  
$texto = trim($texto);  
  
// imprime "texto com espaços"  
print $texto;
```

## ► Sintaxe

### ► Funções Internas – Strings

### ► Genéricas

#### ► **strlen**

- Retorna um inteiro que indica o tamanho de um texto.

```
$texto = "meu texto qualquer";
```

```
print strlen($texto); // imprime 18
```

## ► Sintaxe

### ► Funções Internas – Strings

### ► Genéricas

#### ► **strpos**

- Retorna a posição da primeira ocorrência de um texto dentro de outro.

```
$texto = “maria antonia augusta de oliveira”  
$pos = strpos($texto, “augusta”)
```

```
print $pos; // imprime 15
```

## ► Sintaxe

### ► Funções Internas – Strings

### ► Genéricas

#### ► **substr**

- ❑ Retorna parte de um texto.
- ❑ Argumentos
  - ❑ 1º : Texto
  - ❑ 2º : Posição Inicial
  - ❑ 3º : Tamanho do texto extraído
- ❑ Valores negativos iniciam o recorte do final

```
$texto = "abcdefghij";  
  
print substr($texto, 5) . "<br>";  
print substr($texto, 6, 4) . "<br>";  
print substr($texto, -1) . "<br>";  
print substr($texto, -3, 3) . "<br>";
```

```
fg hij  
gh ij  
j  
hi j
```

## ► Sintaxe

### ► Funções Internas – Strings

### ► Genéricas

#### ► **str\_replace**

- Substitui um conjunto de caracteres dentro de uma string

```
$texto = "Eu vou para a escola";  
  
// imprime "Eu não vou para a escola"  
print str_replace("vou", "não vou", $texto);  
  
// imprime "Você vai para a escola"  
print str_ireplace("eu vou", "Você vai", $texto);
```

#### ► **str\_ireplace**

- Semelhante ao *str\_replace*, só que não faz distinção entre letras maiúsculas e letras minúsculas.

## ▶ Exercício 2

- ▶ Crie uma função que receba uma data no formato “dd/mm/yyyy” e imprima o dia, o mês e o ano de forma isolada.

- ▶ *printDate(“02/10/2015”)*

- ☐ *Dia: 02*

- ☐ *Mês: 10*

- ☐ *Ano: 2015*

## ► Sintaxe

### ► Arrays

- Criado com o construtor [array\(\)](#)

```
$vetor = array("x", "y", "z", "w");  
print_r($vetor);
```

```
Array ( [0] => x [1] => y [2] => z [3] => w )
```

```
echo "Tamanho do Vetor: " . sizeof($vetor);
```

## ► Sintaxe

### ► Arrays

- Reconhece pares separados por vírgula **chave => valor**.

```
$vetor = array(0=>"a", 1=>"b", 2=>"c", 3=>"d");  
print_r($vetor);
```

```
Array ( [0] => a [1] => b [2] => c [3] => d )
```



## ► Sintaxe

### ► Arrays

#### ► Bidimensional

```
$matriz = array(0=>array(0=>"a", 1=>"b"),  
               1=>array(0=>"c", 1=>"d"));  
  
echo "matriz[0][0] = ".$matriz[0][0]."<br>";  
echo "matriz[0][1] = ".$matriz[0][1]."<br>";  
echo "matriz[1][0] = ".$matriz[1][0]."<br>";  
echo "matriz[1][1] = ".$matriz[1][1]."<br>";
```

## ► Sintaxe

### ► Funções Internas – Arrays

### ► Genéricas

#### ► **in\_array**

- Verifica se um determinado valor existe em alguma posição do array.

```
$array = array('pêra', 'uva', 'melão');  
  
if(in_array('uva', $array)){  
    echo 'Valor encontrado';  
}  
  
//Escreve: "Valor encontrado"
```

## ► Sintaxe

### ► Funções Internas – Arrays

### ► Genéricas

#### ► **array\_count\_values**

- Retorna um novo array contabilizando a frequência de cada ocorrência única.

- Chave: ocorrência ; Valor: frequência

```
$array = array('laranja', 'banana', 'maçã', 'laranja');  
print_r(array_count_values($array));  
/* Retorna  
(  
    [laranja] => 2  
    [banana] => 1  
    [maçã] => 1  
)  
*/
```

## ► Sintaxe

### ► Funções Internas – Arrays

### ► Genéricas

#### ► **array\_merge**

- Agrega o conteúdo dos dois arrays passados como parâmetro e gera um novo array com todos os valores.

```
$array1 = array('maça', 'melão', 'goiaba');  
$array2 = array('laranja', 'banana', 'melancia');  
  
print_r(array_merge($array1, $array2));  
/* Retorna  
(  
    [0] => maçã  
    [1] => melão  
    [2] => goiaba  
    [3] => laranja  
    [4] => banana  
    [5] => melancia  
)  
*/
```

## ► Sintaxe

### ► Funções Internas – Arrays

### ► Genéricas

#### ► **array\_combine**

- Mescla os dois arrays e gera um novo array onde as chaves serão os valores do primeiro array e os valores os valores do segundo array.

```
$keys = array('nome', 'email', 'site');  
$values = array('Rafael', 'rafaelwendel@hotmail.com', 'www.rafaelwendel.com');  
print_r(array_combine($keys, $values));  
/* Retorna  
(  
    [nome] => Rafael  
    [email] => rafaelwendel@hotmail.com  
    [site] => www.rafaelwendel.com  
)  
*/
```

## ► Sintaxe

### ► Funções Internas – Arrays

### ► Genéricas

#### ► **explode**

- Divide um texto em pedaços, produzindo um *array* como resultado da operação.

```
$texto = "item1;item2;item3;item4;item5";  
$pedacos = explode(";", $texto);  
  
print_r($pedacos);
```

```
Array  
(  
    [0] => item1  
    [1] => item2  
    [2] => item3  
    [3] => item4  
    [4] => item5  
)
```

## ► Sintaxe

### ► Funções Internas – Arrays

### ► Genéricas

#### ► **implode**

- Junta todos os elementos de um *array* num único texto.

```
$lista = array("uva", "melancia", "banana", "maçã");  
$texto = implode(" - ", $lista);  
  
// imprime "uva - melancia - banana - maçã"  
print $texto;
```

### ▶ Exercício 3

- ▶ Crie uma função que receba o nome completo de uma pessoa.
- ▶ A função deve retornar o nome e o sobrenome da pessoa.
  - ▶ *nomeSobrenome("Joyce Miranda dos Santos")*
  - ▶ *Joyce Santos*



## ► Sintaxe

### ► Estruturas de controle

#### ► IF

```
<?php
if ($a > $b) {
    echo "a is bigger than b";
    $b = $a;
}
?>
```

## ► Sintaxe

### ► Estruturas de controle

#### ► ELSE

```
<?php
if ($a > $b) {
    echo "a is greater than b";
} else {
    echo "a is NOT greater than b";
}
?>
```

## ► Sintaxe

### ► Estruturas de controle

#### ► ELSE IF / ELSEIF

```
<?php
if ($a > $b) {
    echo "a is bigger than b";
} elseif ($a == $b) {
    echo "a is equal to b";
} else {
    echo "a is smaller than b";
}
?>
```

```
if($a > $b):
    echo $a." is greater than ".$b;
elseif($a == $b): // Note the combination of the words.
    echo $a." equals ".$b;
else:
    echo $a." is neither greater than or equal to ".$b;
endif;
```

## ► Sintaxe

### ► Estruturas de repetição

#### ► WHILE

```
$i = 1;
while ($i <= 10) {
    echo $i++; /* the printed value would be
               $i before the increment
               (post-increment) */
}
```

```
$i = 1;
while ($i <= 10):
    echo $i;
    $i++;
endwhile;
?>
```

## ► Sintaxe

### ► Estruturas de repetição

#### ► WHILE

```
$arr = array("orange", "banana", "apple", "raspberry");  
  
$i = 0;  
while ($i < count($arr)) {  
    $a = $arr[$i];  
    echo $a . "\n";  
    $i++;  
}
```

## ► Sintaxe

### ► Estruturas de repetição

#### ► DO WHILE

```
<?php
$i = 0;
do {
    echo $i;
} while ($i > 0);
?>
```

## ► Sintaxe

### ► Estruturas de repetição

#### ► FOR

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

## ► Sintaxe

### ► Estruturas de repetição

#### ► FOREACH

- Forma fácil de iterar sobre arrays

```
foreach (array_expression as $value)  
    statement
```

```
foreach (array_expression as $key => $value)  
    statement
```



## ► Sintaxe

### ► Estruturas de repetição

#### ► FOREACH

```
$arr = array(1,2,3,4);  
foreach($arr as $value) {  
    | echo $value . "<br>";  
}
```

## ► Sintaxe

### ► Estruturas de repetição

#### ► FOREACH

```
$arr = array(1,2,3,4);  
foreach($arr as $chave => $value) {  
    echo $chave . " : " . $value . "<br>";  
}
```

## ► Sintaxe

### ► BREAK

- Encerra a execução de um laço de repetição

### ► CONTINUE

- Ignora o resto da iteração do loop e continua a execução a partir da próxima iteração do loop.

```
$stack = array('first', 'second', 'third', 'fourth', 'fifth');  
  
foreach($stack AS $v){  
    if($v == 'second')continue;  
    if($v == 'fourth')break;  
    echo $v.'<br>';  
}
```

## ► Sintaxe

### ► INCLUDE

- Insere o conteúdo de um arquivo PHP dentro de outro arquivo PHP.
- No caso de erro, NÃO GERA interrupção da execução do script.

```
vars.php
<?php

$color = 'green';
$fruit = 'apple';

?>
```

```
test.php
<?php

echo "A $color $fruit"; // A

include 'vars.php';

echo "A $color $fruit"; // A green apple

?>
```

## ► Sintaxe

### ► REQUIRE

- Insere o conteúdo de um arquivo PHP dentro de outro arquivo PHP.
- No caso de erro, GERA interrupção da execução do script.

```
<?php  
require 'somefile.php';  
?>
```

## ▶ Exercício 4

- ▶ Crie uma função que receba um *array* de produtos.
- ▶ A função deve contabilizar a frequência de cada produto e retornar o produto de maior frequência.
  - ▶ *\$carrinho = array("radio", "tv", "geladeira", "tv");*
  - ▶ *produtoMaiorFreq(\$carrinho)*
  - ▶ *TV: 2 unidades*

## ► Orientação a Objetos

### ► Classes e Objetos

```
<?php
class Produto
{
    // atributos
    public $codigo;

    public $nome;

    public $preco;

    // métodos
    public function calculaDesconto($porcento)
    {
        return $this->preco * $porcento/100;
    }
}
?>
```

**Produto.php**

Produto
- codigo : int - nome : String - preco : double
+ calculaDesconto(porcento : double) : double

```
<?php

include("Produto.php");

$produto = new Produto();
$produto->preco = 1200.00;
echo $produto->calculaDesconto(10);
// imprime 120

?>
```

**TestaProduto.php**

**Quadrado.php**

```
<?php
class Quadrado{

    private $lado;

    //construtor da classe
    function __construct ($lado){
        $this->lado = $lado;
    }

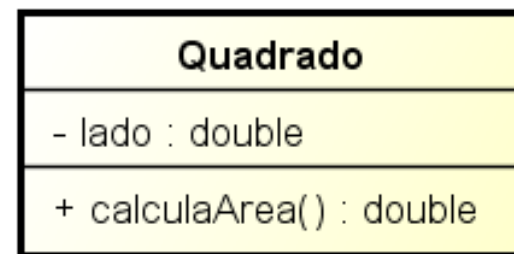
    function getLado(){
        return $this->lado;
    }

    function setLado($lado){
        $this->lado = $lado;
    }

    function calculaArea(){
        return $this->lado * $this->lado;
    }
}
?>
```

## ► Orientação a Objetos

### ► Encapsulamento



**TestaQuadrado.php**

```
<?php

include("Quadrado.php");

$squad = new Quadrado(10.0);

echo " A área do Quadrado de lado " . $squad->getLado() .
      " é " . $squad->calculaArea();

?>
```



## ► Exercício 5

- Crie a classe conforme a modelagem abaixo.
- Crie um objeto da classe, defina seus atributos e execute os métodos.
  - Ano no PHP: *date("Y")*

