



Instituto Federal do Rio Grande do Sul – Rio Grande
Análise e Desenvolvimento de Sistemas
Desenvolvimento Web I

PHP

Prof. Guilherme Vaz Pereira

guilherme.pereira@riogrande.ifrs.edu.br

PHP - INTRODUÇÃO

2

- Linguagem para desenvolvimento de aplicações web;
- *Open Source*
- Linguagem *interpretada*.

Linguagem COMPILADA x INTERPRETADA

3

→ Problema

- ◆ Programa escrito em uma linguagem de alto nível precisa ser traduzido para linguagem de máquina para ser utilizado em um computador.

Linguagem COMPILADA x INTERPRETADA

4

→ Problema

- ◆ Programa escrito em uma linguagem de alto nível precisa ser traduzido para linguagem de máquina para ser utilizado em um computador.

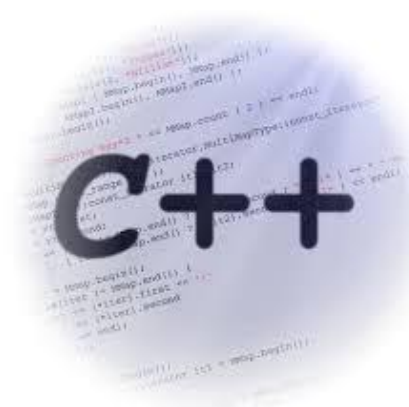
Linguagem COMPILADA x INTERPRETADA

5

→ Linguagem **compilada**

- ◆ Programa (código-fonte) inteiro é traduzido para linguagem de máquina e então carregado para ser executado diretamente no computador.

- ◆ Ex:



Linguagem COMPILADA x INTERPRETADA

6

→ Linguagem **compilada**

- ◆ **Código-Fonte:** escrito na linguagem de alto nível (Ex: JAVA);
- ◆ **Compilador:** Analisa o código fonte sintática e semanticamente para transformar o código de alto nível em código máquina. É o compilador que vai detectar os erros que, eventualmente, tenha cometido;
- ◆ **Programa Executável:** resultado da transformação, pelo compilador, do código fonte em código máquina.

Linguagem COMPILADA x INTERPRETADA

7

→ Linguagem **interpretada**

- ◆ Interpretação é outra maneira de executar uma linguagem de alto nível;
- ◆ A ideia é interpretar diretamente as frases do programa fonte;
- ◆ Programa é interpretado por outro programa, chamado interpretador;

Linguagem COMPILADA x INTERPRETADA

8

→ Linguagem **interpretada**

- ◆ Os interpretadores processam diretamente o programa na linguagem fonte ***e os dados de entrada*** para produzir os seus resultados ou efeitos;
- ◆ Na interpretação o programa fonte é traduzido e executado instrução a instrução, de modo interativo.

Linguagem COMPILADA x INTERPRETADA

9

→ Linguagem **interpretada**

◆ Ex:.



Linguagem COMPILADA x INTERPRETADA

10

→ Linguagem **híbrida**



- ◆ Código JAVA é compilado para *bytecodes*;
- ◆ *bytecodes* são interpretados;
- ◆ JVM (Java Virtual Machine).

INSTALAÇÃO DO PHP

- Instalação em ambiente Windows
- Instalação em ambiente Linux
- Teste
- Primeiro programa
- Mostrando erros

LINUX

12

- No terminal (Ctrl+Alt+t);
- Arquivos devem ser salvos na pasta *var/www/html*
- *Localhost*

```
sudo apt-get install apache2 php5
```

**Servidor de
Aplicação**

Linguagem

WINDOWS

13

→ Instalar pacote BITNAMI

◆ PHP + MySQL


- **WAMP** - <https://bitnami.com/stack/wamp>

◆ PHP + POSTGRES

- **WAPP** - <https://bitnami.com/stack/wapp>

TESTANDO A INSTALAÇÃO

14

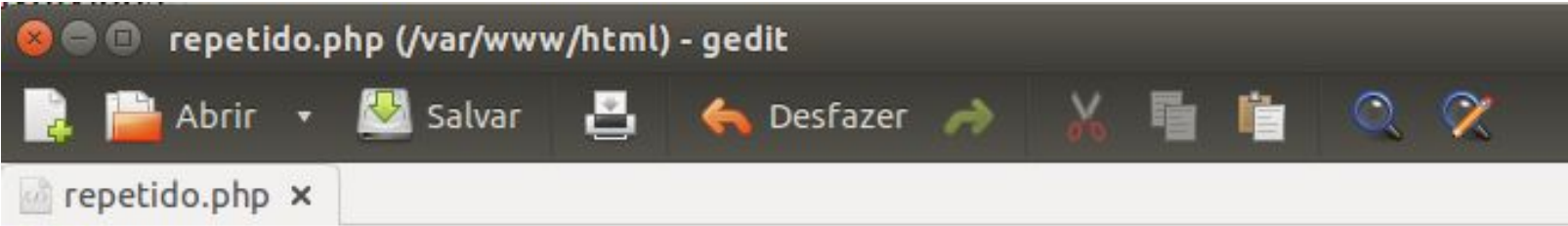


```
<?php  
  
    echo phpinfo();  
  
?>
```

→ <http://localhost/teste.php>

PRIMEIRO PROGRAMA

15



The screenshot shows a gedit editor window titled "repetido.php (/var/www/html) - gedit". The toolbar includes icons for opening files, saving, printing, undo, redo, cut, copy, paste, search, and help. The code editor displays the following PHP code:

```
<?php
    $repetido = true;

    if($repetido){
        echo "Nota zero para todos os envolvidos!!";
    }
    echo "</br></br>";
    echo "Obrigado!!";
?>
```

MOSTRANDO OS ERROS

16

→ No próprio código fonte

```
ini_set('display_errors', 1);  
ini_set('log_errors', 1);  
ini_set('error_log', dirname(__FILE__) . '/error_log.txt');  
error_reporting(E_ALL);
```

→ Editando o arquivo *php.ini*

- ◆ Arquivo `/etc/php5/apache2/php.ini`
- ◆ Alterar a chave `display_errors` de *Off* para *On*

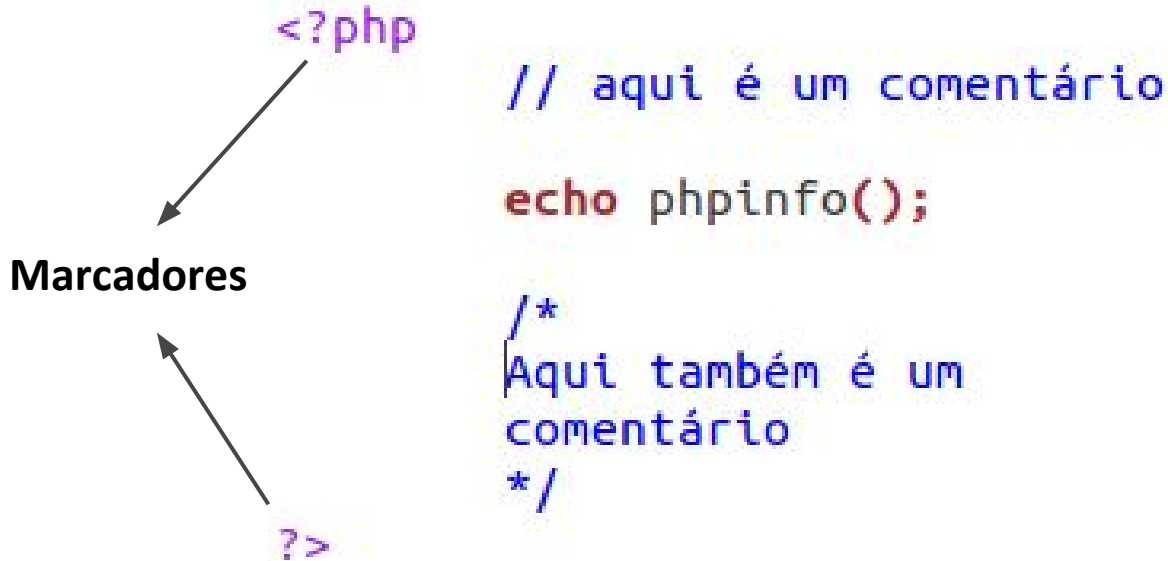
A horizontal bar at the top of the slide, divided into a red section on the left and a green section on the right.

LINGUAGEM PHP

MARCADORES E COMENTÁRIOS

18

→ Marcadores e comentários



VARIÁVEIS

19

- Representadas pelo sinal \$ (cifrão);
- Deve iniciar por uma letra ou pelo símbolo subscrito (_);
- Pode conter letra, número ou subscrito.

```
<?php
    $numero = 1;
    $nome = "Guilherme";
    $_disciplina = "DS I";

    //nome inválido de variável
    $1;

?>
|
```

CONSTANTES

20

Define (<nome>, <valor>, <case insensitive>)

- Por convenção , o nome de uma constante contém somente em letras MAIÚSCULAS;
- Somente tipos escalares (int, strings, boolean ponto flutuante);
- *Case sensitive.*

RESULTADO NO BROWSER

21

```
echo (texto);  
print (texto);
```



TIPOS



TIPOS EM PHP

23

→ Escalares

- ◆ Inteiros (Int);
- ◆ Ponto Flutuante (Float, Double ou Real);
- ◆ String;
- ◆ Booleanos.

TIPOS EM PHP

24

→ Compostos

- ◆ Arrays;
- ◆ Objetos.

→ Especiais

- ◆ Recursos;
- ◆ NULO (Null).

TIPOS EM PHP

25

```
<?php
```

```
$i = 10; // Inteiro  
$nome = "Guilherme"; // String  
$falso = FALSE; // Booleano  
$verdadeiro = TRUE; // Booleano  
$valor = 100.50; // Ponto flutuante
```

```
?>
```

STRINGS

26

- Uma string pode ser definida com aspas simples ou com aspas duplas;

```
<?php
```

```
$disciplina = 'Desenvolvimento de Sistemas I - 2015';
```

```
$disciplina2 = "Desenvolvimento de Sistemas II - 2015";
```

```
?>
```

STRINGS

27

→ **Aspas simples**

- ◆ Se o texto contiver aspas simples, deve ser precedido pelo caractere barra invertida (\);

→ **Aspas duplas**

- ◆ Também é necessário barra invertida para imprimir as aspas duplas;

ARRAY

28

- Arrays Associativos;
- Mapas ordenados por **chave** e **valor**;
- Formas de definição

- ◆ Direta;

```
array([chave =>] valor) .
```

- ◆ Explícita;

```
array[chave] = valor) .
```

- A função `print_r` imprime um vetor.

ARRAY

29

```
<?php  
    $vetor = array();  
    $vetor[1] = 540;  
    $vetor[3] = 8456;  
?>
```

Memória			
posição 0	posição 1	posição 2	posição 3
	540		8456
Espaço de memória	Espaço de memória	Espaço de memória	Espaço de memória

ARRAY

30

→ Explícita

```
1. <?php
2.     $doc = array("rg" => "00.000.00-X",
3.                  "cpf" => "000.000.000-00",
4.                  "cartao de credito" => 12345);
5. ?>
```

→ Direta

```
1. <?php
2.     $doc = array();
3.     $doc['rg'] = "00.000.000-X";
4.     $doc['cpf'] = "000.000.000-00";
5.     $doc['cartao de credito'] = 12345;
6. ?>
```

ARRAY MULTIDIMENSIONAL

31

```
01. <?php
02. $m = array("Fulano" => array("rg" => "00.000.000-1", "cpf" => "000.000.000-01"),
03.         "Ciclano" => array("rg" => "10.100.100-X", "cpf" => "100.100.100-01"),
04.         "Beltrano" => array("rg" => "11.111.111-1", "cpf" => "111.111.111-01"));
05. echo("Pessoas cadastradas..." .
06.
07.     "<BR><BR>Fulano: " .
08.     "<BR>RG: " . $m['Fulano']['rg'] .
09.     "<BR>CPF: " . $m['Fulano']['cpf'] .
10.
11.     "<BR><BR>Ciclano: " .
12.     "<BR>RG: " . $m['Ciclano']['rg'] .
13.     "<BR>CPF: " . $m['Ciclano']['cpf'] .
14.
15.     "<BR><BR>Beltrano: " .
16.     "<BR>RG: " . $m['Beltrano']['rg'] .
17.     "<BR>CPF: " . $m['Beltrano']['cpf']);
18. ?>
```

A horizontal bar spanning the width of the slide, divided into a red left section and a green right section.

OPERADORES

OPERADORES ARITMÉTICOS

33

Exemplo	Nome	Resultado
$-\$a$	Negação	Oposto de $\$a$.
$\$a + \b	Adição	Soma de $\$a$ e $\$b$.
$\$a - \b	Subtração	Diferença entre $\$a$ e $\$b$.
$\$a * \b	Multiplicação	Produto de $\$a$ e $\$b$.
$\$a / \b	Divisão	Quociente de $\$a$ por $\$b$.
$\$a \% \b	Módulo	Resto de $\$a$ dividido por $\$b$.

OPERADORES DE ATRIBUIÇÃO

34

Assignment	Same as:	
<code>\$a += \$b</code>	<code>\$a = \$a + \$b</code>	Addition
<code>\$a -= \$b</code>	<code>\$a = \$a - \$b</code>	Subtraction
<code>\$a *= \$b</code>	<code>\$a = \$a * \$b</code>	Multiplication
<code>\$a /= \$b</code>	<code>\$a = \$a / \$b</code>	Division
<code>\$a %= \$b</code>	<code>\$a = \$a % \$b</code>	Modulus

OPERADORES DE COMPARAÇÃO

35

Exemplo	Nome	Resultado
<code>\$a == \$b</code>	Igual	Verdadeiro (TRUE) se <code>\$a</code> é igual a <code>\$b</code> .
<code>\$a === \$b</code>	Idêntico	Verdadeiro (TRUE) se <code>\$a</code> é igual a <code>\$b</code> , e eles são do mesmo tipo (introduzido no PHP4).
<code>\$a != \$b</code>	Diferente	Verdadeiro se <code>\$a</code> não é igual a <code>\$b</code> .
<code>\$a <> \$b</code>	Diferente	Verdadeiro se <code>\$a</code> não é igual a <code>\$b</code> .
<code>\$a !== \$b</code>	Não idêntico	Verdadeiro se <code>\$a</code> não é igual a <code>\$b</code> , ou eles não são do mesmo tipo (introduzido no PHP4).
<code>\$a < \$b</code>	Menor que	Verdadeiro se <code>\$a</code> é estritamente menor que <code>\$b</code> .
<code>\$a > \$b</code>	Maior que	Verdadeiro se <code>\$a</code> é estritamente maior que <code>\$b</code> .
<code>\$a <= \$b</code>	Menor ou igual	Verdadeiro se <code>\$a</code> é menor ou igual a <code>\$b</code> .
<code>\$a >= \$b</code>	Maior ou igual	Verdadeiro se <code>\$a</code> é maior ou igual a <code>\$b</code> .

OPERADORES DE ARRAY

36

Exemplo	Nome	Resultado
<code>\$a + \$b</code>	União	União de <code>\$a</code> e <code>\$b</code> .
<code>\$a == \$b</code>	Igualdade	TRUE se <code>\$a</code> e <code>\$b</code> tem os mesmos pares de chave/valor.
<code>\$a === \$b</code>	Identidade	TRUE se <code>\$a</code> e <code>\$b</code> tem os mesmos pares de chave/valor na mesma ordem e do mesmo tipo.
<code>\$a != \$b</code>	Desigualdade	TRUE se <code>\$a</code> não é igual a <code>\$b</code> .
<code>\$a <> \$b</code>	Desigualdade	TRUE se <code>\$a</code> não é igual a <code>\$b</code> .
<code>\$a !== \$b</code>	Não identidade	TRUE se <code>\$a</code> não é identico a <code>\$b</code> .

Mais operadores em: https://php.net/manual/pt_BR/language.operators.php

A horizontal bar at the top of the slide, divided into a red section on the left and a green section on the right.

ESTRUTURAS DE CONTROLE

ESTRUTURAS DE CONTROLE

38

→ IF - ELSE

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

ESTRUTURAS DE CONTROLE

39

→ WHILE

```
<?php
/* example 1 */

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

ESTRUTURAS DE CONTROLE

40

→ FOREACH

```
<?php
/* foreach example 1: value only */

$a = array(1, 2, 3, 17);

foreach ($a as $v) {
    echo "Current value of \$a: $v.\n";
}
```


ESTRUTURAS DE CONTROLE

41

→ FOREACH

```
/* foreach example 3: key and value */
```

```
$a = array(  
    "one" => 1,  
    "two" => 2,  
    "three" => 3,  
    "seventeen" => 17  
);  
  
foreach ($a as $k => $v) {  
    echo "\$a[$k] => $v.\n";  
}
```

ESTRUTURAS DE CONTROLE

42

→ FOREACH

```
/* foreach example 4: multi-dimensional arrays */  
$a = array();  
$a[0][0] = "a";  
$a[0][1] = "b";  
$a[1][0] = "y";  
$a[1][1] = "z";  
  
foreach ($a as $v1) {  
    foreach ($v1 as $v2) {  
        echo "$v2\n";  
    }  
}
```

ESTRUTURAS DE CONTROLE

43

→ CONTINUE

- ◆ “Pula” iteração atual.

→ BREAK

- ◆ Finaliza a instrução dentro de uma estrutura de controle.

ESTRUTURAS DE CONTROLE

44

```
<?php
    $cont = 0;

    echo "Break";
    echo "<br>";

    while ($cont < 10) {
        if ($cont == 5) {
            break;
        }
        echo $cont;
        $cont++;
    }
?>
```

```
<?php
    $cont = 0;

    echo "Continue";
    echo "<br>";

    while ($cont<10) {
        if ($cont == 5) {
            continue;
        }
        echo $$cont;
        $cont++;
    }
?>
```

ESTRUTURAS DE CONTROLE

45

→ SWITCH

```
switch ($i) {  
    case 0:  
        echo "i equals 0";  
        break;  
    case 1:  
        echo "i equals 1";  
        break;  
    case 2:  
        echo "i equals 2";  
        break;  
}
```

ESTRUTURAS DE CONTROLE

46

→ REQUIRE e INCLUDE

- ◆ Têm por objetivo inserir pedaços de código php no script atual;
- ◆ A diferença está em como tratam os erros;
- ◆ Include produz um warning;
- ◆ Require produz um erro fatal;
- ◆ `include_once` e `require_once`.

ESTRUTURAS DE CONTROLE

47

→ REQUIRE e INCLUDE

- ◆ Têm por objetivo inserir pedaços de código php no script atual;
- ◆ A diferença está em como tratam os erros;
- ◆ Include produz um warning;
- ◆ Require produz um erro fatal;
- ◆ `include_once` e `require_once`.



FUNÇÕES



FUNÇÕES

49

```
<?php
function foo ($arg_1, $arg_2, /* ..., */ $arg_n)
{
    echo "Exemplo de função.\n";
    return $valor_retornado;
}
?>
```

FUNÇÕES

50

- Passagem de parâmetro por referência;
- Permite que a função modifique seus parâmetros.

```
<?php
function add_some_extra(&$string)
{
    $string .= ' e alguma coisa mais.';
}
$str = 'Isto é uma string,';
add_some_extra($str);
echo $str;    // imprime 'Isto é uma string, e alguma coisa mais.'
?>
```

FUNÇÕES

51

→ Funções úteis

- ◆ ***func_num_args*** - retorna quantos argumentos a função recebeu;
- ◆ ***func_get_arg*** - retorna um argumento da função;
- ◆ ***func_get_args*** - retorna um array com todos os argumentos de uma função.

Referências

52

- Manual PHP: https://php.net/manual/pt_BR/index.php
- Soares, Wallace. PHP 5: Conceitos, Programação e Integração com Banco de Dados -- 6 ed. rev., atual. -- São Paulo: Érica, 2010.