



**DIO**  
CEZAR  
**GO**

# PHP

## parte 1



# O QUE É?

- PHP: PHP HyperText Preprocessor;
- linguagem de código fonte aberto;
- muito utilizada na *web* → PHP is used by 82.8% of all the websites whose server-side programming language we know.
  - <https://goo.gl/5WuCkc>
- criada para desenvolvimento de aplicações *web*;

# DIFERENCIAL

---

- diferencial → ao invés de criar um programa com comandos para imprimir HTML (Java?), escreve-se um arquivo “HTML” com um código específico que faz alguma coisa;
- como? → o código PHP é delimitado por tags iniciais e finais que lhe permitem pular para dentro e para fora do “modo PHP”;
  - dentro do modo PHP, têm-se todos os recursos oferecidos pela linguagem, como: acesso a base de dados, manipulação de imagens, entre outros.

# **VANTAGEM**

---

a melhor coisa em usar PHP está no fato de ele ser extremamente simples para um iniciante, mas oferecer muitos recursos para o programador profissional

# CONFIGURANDO O SEU AMBIENTE

---

- existem diferentes formas de se configurar um ambiente de desenvolvimento;
- a configuração básica necessária é: um servidor web + compilador da linguagem php;
- o compilador essencialmente é o programa responsável por interpretar as instruções inseridas em nossos códigos;
  - este compilador deve ser instalado correspondente ao sistema operacional utilizado;
- quando o compilador estiver instalado, deve-se fazer sua “ligação” com o servidor *web* escolhido;

# CONFIGURANDO O SEU AMBIENTE

---

- para utilizar o PHP, deve-se utilizar um servidor com devido suporte;
- em relação aos servidores temos os *bigplayers*:
  - <https://www.apache.org/>
  - <https://www.nginx.org/>

# CONFIGURANDO O SEU AMBIENTE

---

- cada um tem sua especificidade, vantagens e desvantagens;
- isso é uma das... → tretas da internet:
  - Apache vs NGINX → <https://goo.gl/A2JpSD>

# CONFIGURANDO O SEU AMBIENTE

---

- temos que escolher um deles... então... apache!



# SERVIDOR APACHE

---

- o servidor web Apache é largamente utilizado no mundo todo;
- esta liderança deve-se ao fato de ter um excelente desempenho, alto nível de personalização, confiabilidade, portabilidade, vasta documentação disponível e seu baixo custo (free);
- a palavra Apache significa A PAtCHy, pois foi baseado em um código juntamente com uma série de arquivos patch;
- para muitos desenvolvedores, porém, a palavra faz referência aos nativos americanos, ou seja, os índios Apache.

# SERVIDOR APACHE

---

- altamente configurável, pode ser executado em diferentes plataformas;
- flexível, está sempre em desenvolvimento para a inclusão dos protocolos mais atualizados;
- fornece o código-fonte completo e não possui licenças restritivas;
- pode ser configurado para diferentes funções;
- é composto de módulos, cada um implementando uma característica diferente e aumentando a funcionalidade do servidor;

# SERVIDOR APACHE

---

- a configuração do Apache é toda feita sob um diretório:  
.../apache[x.x.x]/conf/
- o arquivo mais importante a ser configurado é o httpd.conf, que contém as configurações de administração de servidor;

# SERVIDOR APACHE

---

- dentre algumas configurações podemos destacar:
  - ServerName: nome do servidor HTTP;
  - DocumentRoot: é o diretório em que os arquivos das páginas serão armazenados;
  - ServerRoot: configura o diretório base (topo) sob o qual os arquivos de log, configuração e de erros são mantidos;
  - DirectoryIndex: ajusta o arquivo que o Apache chamará, quando o diretório principal for requisitado.

# INSTALANDO E CONFIGURANDO O SEU AMBIENTE

- você vai precisar instalar e configurar o seu ambiente de desenvolvimento/produção;
- seu ambiente de desenvolvimento pode utilizar um software que facilita tudo:
  - XAMPP → [https://www.apachefriends.org/pt\\_br/index.html](https://www.apachefriends.org/pt_br/index.html)
  - WAMP → <http://www.wampserver.com/en/>
  - EasyPHP → <http://www.easyphp.org/>
  - Outros? → <https://goo.gl/bsdnvn>

# **INSTALANDO E CONFIGURANDO O SEU AMBIENTE**

- seu ambiente de produção deve ter um cuidado especial;
  - configurações de segurança;
  - logs de atividades;
  - otimização de performance;
- Digital Ocean For Zombies →  
<https://github.com/diogocezar/dctb-digital-ocean-for-zombies>

# AMBIENTE INSTALADO

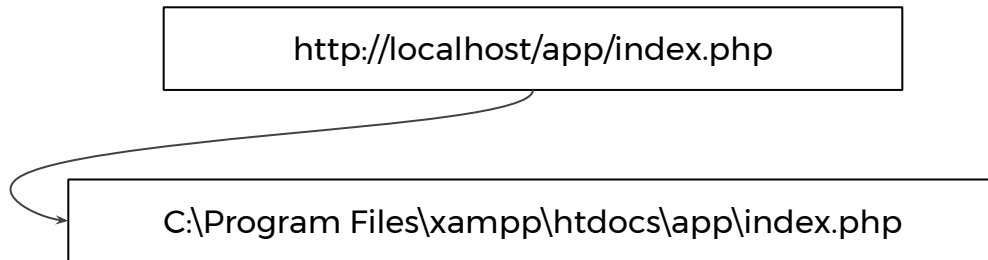
- para nossa disciplina, XAMPP;
  - Apache 2.x;
  - PHP 5.x ou 7.x;
  - MySQL? → MariaDB;
  - PhpMyAdmin;

algumas novidades aqui:

<https://tableless.com.br/10-novidades-do-php-7/>

# OLÁ MUNDO

- você pode utilizar seu editor de textos de preferência, ou uma IDE específica, para editar seus arquivos .php;
- as páginas PHP devem ser salvas no diretório raiz do servidor, e serão executadas somente quando acessadas pela *url* correspondente em seu navegador;





# OLÁ MUNDO

para criar o primeiro exemplo, digite o seguinte código-fonte no seu editor e salve com o nome de teste.php dentro do diretório raiz do servidor:

## OLÁ MUNDO

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <?php echo "<p>Olá Mundo</p>"; ?>
  </body>
</html>
```

# OLÁ MUNDO

- em seu navegador, digite:
  - <http://127.0.0.1/teste.php>
  - <http://localhost/teste.php>
- analise também o seu código fonte → exibir → código fonte;
  - note que os comandos em php não aparece em seu código fonte;
  - o servidor interpreta todos os comandos php antes de enviar a página pronta para o seu navegador;

# O QUE SE PODE FAZER COM PHP?

- aplicações server-side;
  - a mais tradicional;
  - você precisa de três coisas para seu trabalho:
    - o interpretador do PHP (como CGI ou módulo);
    - um servidor web;
    - um browser;
  - basta rodar o servidor web conectado a um PHP instalado, assim você pode acessar os resultados de seu programa PHP com um browser, visualizando a página PHP através do servidor web.

# O QUE SE PODE FAZER COM PHP?

- script de linha de comando:
  - você pode fazer um *script* PHP funcionar sem um servidor *web* ou navegador;
  - utilizando apenas o seu interpretador → exemplo: `$php myscript.php`
  - usados na maioria das situações para execução de serviços locais;

# O QUE SE PODE FAZER COM PHP?

- aplicações GUI;
  - o PHP não é (provavelmente) a melhor linguagem para produção de aplicações com interfaces em janelas;
  - PHP-GTK;

# O QUE SE PODE FAZER COM PHP?

- o PHP pode ser utilizado na maioria dos sistemas operacionais:
  - Linux;
  - várias variantes Unix (incluindo HP-UX, Solaris e OpenBSD);
  - Microsoft Windows;
  - Mac OS X;
  - e outros...

# **O QUE SE PODE FAZER COM PHP?**

---

- o PHP também é suportado pela maioria dos servidores web atuais:
  - Apache;
  - Microsoft Internet Information Server;
  - Personal Web Server;
  - Netscape and iPlanet Servers;
  - O'Reilly Website Pro Server;
  - Caudium;
  - Xitami;
  - OmniHTTPd;
  - e outros...

# O QUE SE PODE FAZER COM PHP?

- diversidade
  - com o PHP, você tem a liberdade para escolher o sistema operacional e o servidor web, bem como, escolher entre utilizar programação estrutural ou programação orientada a objeto, ou ainda uma mistura deles;



# O QUE SE PODE FAZER COM PHP?

- e ainda... com PHP você não está limitado a gerar somente HTML. As habilidades do PHP incluem:
  - geração de imagens;
  - geração de arquivos pdf;
  - entre várias libs específicas para trabalhar com uma infinidade de recursos;

# **O QUE SE PODE FAZER COM PHP?**

- suporte a múltiplos bancos de dados, através de conexões nativas e/ou por plugins de conexão;
  - PDO;

# CONFIGURAÇÃO DO PHP

---

- as configurações do PHP ficam armazenadas em um arquivo denominado php.ini e que é carregado cada vez que o servidor é iniciado;
- no Windows, ele fica na pasta c:\Windows; (quando nativamente instalado)
- através de modificações neste arquivo é possível alterar várias opções no comportamento do PHP;
  - inclusive a instalação e/ou inclusão de novas bibliotecas nativas;
  - todas as linhas iniciadas por ponto-e-vírgula são comentários;
- ao utilizar a ferramenta XAMPP essa configuração está disponível em PHP → Xampp\php\php.ini.

# SINTAXE BÁSICA

---

- tags especiais indicam ao PHP onde estão os blocos de código;
- a tag de abertura é formada por um sinal de “menor que” (<), um sinal de interrogação (?) e a sigla php → <?php
- a tag de fechamento é formada por um ponto interrogação (?) e sinal de “maior que” (>). → ?>

# PRIMEIRO EXEMPLO

---

## PRIMEIRO EXEMPLO DE PHP

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <?php
      $a = 10;
      $b = 15;
      $c = $a + $b;
      echo "$a mais $b é igual a $c";
    ?>
  </body>
</html>
```

# PRIMEIRO EXEMPLO

---

- sinal de ponto-e-vírgula (;) indica o final de um comando;
- note que, neste caso, o código php é inserido dentro de um código HTML;
- execute o script e veja qual será seu resultado;

# BLOCOS DE ABERTURA E FECHAMENTO

---

## PRIMEIRO EXEMPLO DE PHP

```
...
<body>
    <?php
        $a = 10;
        if($a > 10){
            ?>
            <h1>A variável é maior que 10</h1>
            <?php
                } else {
                ?>
                <h1>A variável é menor que 10</h1>
                <?php
                    }
                ?>
        </body>
</html>
```

# COMENTÁRIOS

---

- os comentários de mais de uma linha no PHP são obtidos através de `/*` e `*/`;
- os comentários de apenas uma linha são obtidos através de `//` ou `#`;
- os comentários não aparecem no *browser*.



# PALAVRAS RESERVADAS

---

and	var	switch
do	case	xor
for	elseif	continue
include	function	false
require	new	if
true	static	or
break	virtual	this
else	class	while
foreach	extends	default
list	global	
return	not	

# VARIÁVEIS

---

- variáveis armazenam valores. pode-se referir a variáveis para obter seu valor ou para alterar seu conteúdo;
- no PHP elas são representadas por um cifrão (\$) mais o nome da variável;  
exemplo → \$userId;
- os nomes de variáveis válidos são iniciados por letras ou por um subscrito (\_);  
exemplo → \$num, \$\_idade;
- existe diferenciação entre nomes de variáveis maiúsculas e minúsculas;  
exemplo → \$a, \$A, \$\_A, \$\_a.

# EXEMPLO VARIÁVEL

---

## EXEMPLO VARIÁVEL

```
<?php
    $a = 10;
    $A = 20;
    echo "O valor de '$a' é $a e o de '$A' é $A";
?>
```

## RESULTADO

O valor de \$a é 10 e o valor de \$A é 20

# ESCOPO DAS VARIÁVEIS

---

- as variáveis criadas dentro de uma função (function) valerão somente enquanto aquela função existir;
- para trabalhar com uma variável externa à função devemos definir a variável como global;

# EXEMPLO ESCOPO DE VARIÁVEL

---

## EXEMPLO VARIÁVEL

```
<?php
    function soma($a, $b){
        global $msg;
        return $msg.($a + $b);
    }
    $msg = "A soma é ";
    echo soma(10, 2);
?>
```

## RESULTADO

A soma é 12

# INTERPRETAÇÃO DE VARIÁVEIS

- é comum utilizarmos uma mistura de *strings* e variáveis para imprimir o conteúdo desejado;
- têm-se dois tipos de sintaxe:
  - a sintaxe simples;
  - a sintaxe complexa;

# SINTAXE SIMPLES EM STRINGS

---

- sintaxe simples:
  - se um sinal de cifrão (\$) é encontrado, o interpretador tentará obter vários identificadores para formar um nome válido para aquela variável;
  - envolva o nome da variável com chaves {} para especificar explicitamente o fim do nome;

# SINTAXE SIMPLES EM STRINGS

---

## EXEMPLO SINTAXE SIMPLES

```
<?php
    $cerveja = 'Skol';
    echo "Ele bebeu algumas $cervejas"; // não funciona
    echo "Ele bebeu algumas ${cerveja}s"; // funciona
    echo "Ele bebeu algumas {$cerveja}s"; // funciona
?>
```



# SINTAXE COMPLEXA EM STRINGS

---

- sintaxe complexa:
  - é chamada de sintaxe complexa porque você pode incluir expressões complexas dessa maneira;
  - basta escrever a expressão da mesma maneira que faria fora da string, e posteriormente envolvê-la por chaves {};

# SINTAXE COMPLEXA EM STRINGS

---

## EXEMPLO SINTAXE COMPLEXA

```
<?php
    echo "Imprimindo elemento do array : {$myArr[0][0]}";
    echo "Imprimindo um atributo do objeto : {$myObj->name}";
?>
```

# **ASPAS E CONCATENAÇÃO EM STRINGS**

- variáveis dentro de aspas duplas “” são interpretados e mostram seu valor;
- variáveis dentro de aspas simples ‘’ não são interpretadas, e mostram o que está escrito entre as aspas;
- para concatenar variáveis utiliza-se o caractere ponto “.”;

# ASPAS E CONCATENAÇÃO EM STRINGS

## EXEMPLO DE ASPAS E CONCATENAÇÃO EM STRINGS

```
<?php
    $var1 = "php is";
    $var2 = "awesome";
    echo "$var1"; // imprime conteúdo de $var
    echo '$var1'; // imprime $var;
    echo "$var1" . " " . $var2; // php is awesome;
?>
```

# TIPOS DE DADOS

---

- o PHP suporta vários tipos de dados:
  - inteiro → números inteiros (isto é, números sem ponto decimal);
  - números de dupla precisão → números reais (isto é, números que contêm um ponto decimal);
  - string → texto entre aspas simples ( ' ' ) ou duplas ( " " );
  - booleanos → armazenam valores verdadeiros ou falsos, usados em testes de condições;
  - arrays → grupo de elementos do mesmo tipo;
  - objeto → grupo de atributos e métodos;
  - nulo → nenhum valor;

# TIPOS DE DADOS

---

- o PHP é uma linguagem não tipada;
- o tipo da variável não é configurado pelo programador:
  - isso é decidido em tempo de execução pelo PHP, dependendo do contexto no qual a variável é usada.
- pode-se armazenar valores inteiros, positivos ou negativos.
- pode-se usar também valores hexadecimais;
- para verificar o tipo de dados de uma determinada variável pode utilizar as funções: `is_{type}`; exemplo → `if(is_int(5)) // true`
- uma representação legível de um tipo pode ser obtida pela função `gettype()`;
  - exemplo → `echo gettype($str) // imprime "string"`

# TIPOS DE DADOS

---

- pode-se forçar a conversão de uma variável para um certo tipo;
- utilizando o tradicional casting ou
- usar a função `settype()` nela;

## EXEMPLO TIPOS DE DADOS

```
<?php
    $foo = "5bar";           // string
    $bar = true;             // boolean
    settype($foo, "integer"); // $foo é agora 5 (integer)
    (int) $bar;              // $bar é 1
?>
```

# TIPOS DE DADOS

---

## EXEMPLO TIPOS DE DADOS

```
<?php
    $a = 0x1A; // corresponde ao decimal 26
    $b = -16;
    $c = $a + $b;
    $price = 11.90 // ponto flutuante representado por .
    echo "a + b = $c";
?>
```



# ARRAYS

---

- são variáveis que permitem o armazenamento de diversos elementos de qualquer tipo;
- os tipos inclusive podem ser diferentes em um mesmo array;
- um dos elementos pode ser inclusive um outro array, criando possibilitando arrays multidimensionais;
- é possível definir também os índices;
  - estes índices podem ser strings;
- ao omitir o índice quando informar um novo valor, o PHP entenderá que o novo índice é o valor antigo + 1;
- operador => indica o valor de um elemento do array;

# ARRAYS

---

## EXEMPLO ARRAYS

```
<?php
    $items = array(1 => 10,
                   2 => array(10, 20),
                   'teste' => "beer");
    echo $items[1];
    echo $items['teste'];
?>
```

# CONSTANTES

---

- constantes são identificadores para valores simples;
- o seu conteúdo não muda durante a execução do código;
- são criadas com a função define e, por convenção, são escritas com letras MAIÚSCULAS e não usam o cifrão no início;

## EXEMPLO ARRAYS

```
<?php
    define("HEY", "Hello!");
    echo HEY;
?>
```

# EXPRESSÕES

---

tudo que tem um valor pode ser considerado uma expressão  
uma expressão pode ser utilizada em declarações condicionais

## EXEMPLO EXPRESSÕES

```
<?php
    $b = ($a = 5);
    echo "The value of 'b' is $b";
    if($a > $b)
        echo "a bigger than b"
?>
```

# OPERADORES ARITMÉTICOS

---

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 COMPARAÇÃO

---

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

# OPERADORES LÓGICOS

---

EXEMPLO	NOME	RESULTADO
\$a and \$b	E	Verdadeiro (TRUE) se tanto \$a quanto \$b são verdadeiros.
\$a or \$b	OU	Verdadeiro se \$a ou \$b são verdadeiros.
\$a xor \$b	XOR	Verdadeiro se \$a ou \$b são verdadeiros, mas não ambos.
! \$a	NÃO	Verdadeiro se \$a não é verdadeiro.
\$a && \$b	E	Verdadeiro se tanto \$a quanto \$b são verdadeiros.
\$a    \$b	OU	Verdadeiro se \$a ou \$b são verdadeiros.

# OPERADORES DE INCREMENTO E DECREMENTO

---

EXEMPLO	NOME	EFEITO
++\$a	Pré-incremento	Incrementa \$a em um, e então retorna \$a.
\$a++	Pós-incremento	Retorna \$a, e então incrementa \$a em um.
--\$a	Pré-decremento	Decrementa \$a em um, e então retorna \$a.
\$a--	Pós-decremento	Retorna \$a, e então decrementa \$a em um.



# OPERADOR TERNÁRIO

---

é uma forma abreviada de usar o comando condicional if;

cond ? exp1 : exp2

se a condição for verdadeira, o valor retornado é o da exp1, caso contrário o valor retornado é o da exp2;

## EXEMPLO TERNÁRIO

```
<?php
    $points = ($frequency >= 0.75) ? ($points++) : ($points--);
?>
```


# **PRECEDÊNCIA DE OPERADORES**

a precedência de operadores especifica quem tem maior prioridade quando duas delas estão juntas, por exemplo:


$$1 + 5 * 3$$

a multiplicação tem prioridade sobre a adição, então a resposta é 16 e não 18.

# PRECEDÊNCIA DE OPERADORES



<b>clone new</b>
<b>[</b>
<b>++ --</b>
<b>~ - (int) (float) (string) (array) (object) (bool) @</b>
<b>instanceof</b>
<b>!</b>
<b>* / %</b>
<b>+ - .</b>
<b>&lt;&lt; &gt;&gt;</b>
<b>&lt; &lt;= &gt; &gt;= &lt;&gt;</b>
<b>== != === !==</b>



<b>&amp;</b>
<b>^</b>
<b> </b>
<b>&amp;&amp;</b>
<b>  </b>
<b>? :</b>
<b>= += -= *= /= . = % = &amp; =   = ^ = &lt;&lt; = &gt;&gt; =</b>
<b>and</b>
<b>xor</b>
<b>or</b>
<b>,</b>

# FUNÇÕES ÚTEIS - ARRAYS

- `sort($arr)` → ordena um array;
- `count($arr)` → conta o número de elementos e retorna um inteiro;
- `print_r($arr)` → imprime um array (índices e valores);
- `unset($arr[$i])` → remove um elemento do array;
- `unset($arr)` → limpa todo o array;
- `shuffle($arr)` → mistura os elementos de um array;
- `in_array("valor", $arr)` → checa se um valor existe em um array;

# FUNÇÕES ÚTEIS - STRINGS

- `substr($str, $start, $length)` → retorna a parte de string especificada pelo parâmetro `start` e `length`;
- `strrpos($procurado, $str)` → encontra a última ocorrência de um caractere em uma string;
- `strpos($procurado, $str)` → encontra a posição da primeira ocorrência de uma string;
- `trim($str)` → retira espaço no início e final de uma string;
- `str_replace($str, $procurado, $alterar)` → substitui todas as ocorrências da string de `$procurado` com a string de `$alterar`
- `str_len($str)` → conta o número de caracteres de `$str`;
- `strtolower($str)` → transforma todos caracteres em minúsculos;
- `strtoupper($str)` → transforma todos caracteres em maiúsculos;

# FUNÇÕES ÚTEIS - GERAIS

- `empty($valor)` → verifica se determinado valor está vazio;
- `explode('.', $string)` → recorta a string e a transforma em um array, em todo lugar que o caracter '.' for encontrado;
- `implode('.', $array)` → retorna uma string contendo os elementos da matriz na mesma ordem com uma ligação entre cada elemento;
- `number_format($valor, $numCasasDecimais, $ponto, $virgula)` → formata um número com os milhares agrupados;
- `rand($inicio, $fim)` → gera um inteiro aleatório;
- `date('d/m/Y - H:i:s')` → comando para exibição de data;

# ESTRUTURAS DE CONTROLE

---

- sua sintaxe é bem semelhante ao que já foi visto em C e Java;
- as estruturas são:
  - if / else;
  - switch;
  - while;
  - do while;
  - for;

# ESTRUTURAS DE CONTROLE - FOREACH

---

- é uma estrutura de controle um pouco diferente do for;
- é utilizada para navegar entre os elementos de um array;
- apresenta 2 sintaxes possíveis:

## SINTAXE 1

```
<?php
    foreach($arr as $el){
        // instructions
    }
?>
```

## SINTAXE 2

```
<?php
    foreach($arr as $key => $val){
        // instructions
    }
?>
```



# **MATERIAIS COMPLEMENTARES**

- uma ótima referência → <http://br.phptherightway.com/>