**PHP 7**

**O que é?**

**Hypertext Preprocessor** ou PHP, é uma linguagem de programação focada na criação de aplicações do lado do servidor;

Open source- de uso livre, pode ser baixado gratuitamente em qualquer máquina;

Aplicações:

– Script de linha e comando (para automatização de tarefas), utilizado por exemplo no CMD do Windows;

- Aplicações desktop (PHP - GTK), aplicações que são instaladas no computador que pode ser interpretada com Client-Side;

- Aplicações Server-Side (Scripts), aplicações que rodam do lado do servidor, para criação de páginas Web dinâmicas.

É uma das linguagens de programação mais utilizadas para desenvolvimentos de sites e/ou sistemas Web dinâmicos;

Linguagem de programação interpretada (scripts não são submetidos a compilação prévia), os scripts serão interpretados em tempo de requisição das páginas;

Disponível em diversos sistemas operacionais (como Windows, Linus, OS X e Unix);

Inicialmente ele suportaria o paradigma estrutural, porém com o tempo, o paradigma de Orientação a Objetos foi incorporado. Hoje ele suporta não só a utilização, mas também, a combinação dos dois paradigmas;

Suporte a sistemas gerenciadores de banco de dados;

Suporte a protocolos de comunicações disponíveis na internet.

**Embutindo blocos PHP em páginas HTML**

**Nome Tag**

Tag padrão **<?php** código aqui**?>** Habilitada

Tag de impressão **<?=** código aqui**?>** Habilitada

Tag curta **<?** código aqui**?>** Desabilitada

\*Asp tag **<%** código aqui **%>**

Descontinuada na versão 7

Para utilizar códigos PHP, é necessário encapsular o código entre tags (que são específicas do PHP;

As tags podem ser utilizadas em qualquer local dentro do arquivo.

Na versão 8 a tag (<?...?>) não está mais disponível. Deve utilizar a tag padrão.

Config do PHP

XAMPP>Config>PHP (php.ini)

Localizar: XAMPP>Explorer>php> php.ini

Abre arquivo, CTRL+F para localizar ‘short\_open\_tag=Off’

Muda o ‘Off’ para ‘On’ e salva. Reinicia XAMPP (stop, start)

<?php

    echo 'Utilizando a tag padrão'

    ?>

    <br />

    <?= 'Utilizando a tag impressão' ?>

    <br />

    <? echo 'Utilizando a tag curta' ?>

Única tag que se diferencia é a de impressão que tem objetivo explícito

**Ponto e vírgula é obrigatório**, a não ser que tenha apenas uma instrução como no exemplo acima.

**Saída de dados ‘echo() e print()’**

Possuem objetivo claro – **impressão de conteúdo** do bloco de código **PHP dentro do** arquivo . Basicamente uma janela entre a programação e o arquivo HTML;

<hr/> e <br/> podem ser utilizados.

 <?php

    echo 'Comando echo<br/>'; //construtor da linguagem

    print "Comando print<hr/>"; //inicialmente era uma função (que pode ser utilizada normalmente)

?>

//quando função print é executada corretamente, retorna um, quando há erro, retorna 0

    echo print("comando print<hr/>")//result=1

**Funcionamento do PHP mais a fundo (páginas estáticas x dinâmicas)**

De um lado existe o cliente, onde é executado aplicações:

Browsers ou Navegadores – Chrome, FireFox, Edge, Safari e etc.

Essas aplicações fazem requisições através de um meio físico, a Internet.

A requisição tem que sair do computador e ir até uma outra ponta, que pode ser inclusive o computador se for utilizado o **localhost**.

Na outra ponta, deve existir um servidor HTTP (no nosso caso esse servidor é o Apache), capaz de receber e tratar dessas requisições.

A requisição é feita para um arquivo **.php** em que do lado do servidor haverá uma linguagem de programação que vai atuar sobre essa requisição que está sendo feita para o servidor.

Páginas estáticas: Arquivos HTML com codificação HardCoded, onde o servidor HTTP recupera essa requisição e devolve um arquivo;

Páginas dinâmicas: São produzidas no intervalo entre a requisição e a resposta que é recebida e devolvida pelo servidor HTTP (Apache, nesse caso).

**Comentários**

Usados principalmente para documentar o código.

Quando o código está sendo criado, tudo que está sendo feito fica fresco em nossa mente, logo, não há muita dificuldade para entender os blocos de código. Mas, quando depois de um tempo voltamos para fazer manutenção nesses mesmos códigos, é comum ter dificuldade na interpretação da codificação.

A uma confusão até que o entendimento do código seja estabelecido novamente. Principalmente quando há uma certa complexidade no código.

Com isso, comentar o código se torna extremamente útil para fins de documentação. Auxiliando na manutenção futura do código.

No PHP há três formas de fazer comentário nos códigos:

***Comentário de 1 linha***

    #Comentário de uma linha

***Comentário de 1 linha no estilo shell***

    #Comentário de uma linha

***Comentário de múltiplas linhas***

/\*

            Comentário de múltiplas linhas.

            Este tipo de comentário permite que várias linhas

            sejam comentadas ao mesmo tempo

        \*/

**Variáveis**

***Introdução***

**O que são?**

Utilizadas para armazenamento temporário de dados.

**Tipos de variáveis em PHP**

***string*** - armazenam textos, cadeias de caracteres;

***int*** – armazenam valores numéricos inteiros (positivos e negativos);

ex.: 2, 5, -10

***float*** – armazenam valores numéricos (positivos e negativos), mas, com fração;

ex.: 10.5, -7.4

***boolean*** – armazenam estados (true ou false), podem ser utilizados em testes lógicos;

***array*** – como um mapa ordenado de variáveis, relacionados através de chaves e valores.

**Regras de declaração de variáveis em PHP**

Deve obrigatoriamente iniciar com o caractere **‘$’**;

Não pode conter espaços ou caracteres especiais (com exceção do underline);

Não requer, nem suporta a definição explícita de tipos, tipagem dinâmica;

Case sensitive, faz distinção entre letras minúsculas e maiúsculas.

***$nome $Nome $NOME***

**Exemplos de variáveis válidas e inválidas**

**$nome $fone1 $endereco\_2 $\_nome1**

**$ idade $1fone $número $\* nome\_**

Caracteres numéricos não podem estar no início da palavra.

//string

    $nome = 'Ana Almeida'; //podem ser utilizadas aspas simples ou duplas

    //int

    $idade = 17;

    $estudos = 'Desenvolvimento de sistemas';

    //float

    $peso = 50.5;

    //boolean - simples, armazena estado de true ou false(muito utilizadas em testes lógicos)

    $fumante\_sn = true;//(true=1) ou false = vazio

**Concatenação**

//operador '.'

    echo 'Olá ' . $nome . ', vi que sua cor preferida é ' . $cor . ', que você possui ' . $idade . ' anos e que você gosta de ' . $hobby;

    //aspas duplas - variáveis ficam dentro das aspas

    echo '<br/>';

    echo "Olá $nome, vi que sua cor preferida é $cor, que você possui $idade anos e que você gosta de $hobby"

**Variáveis constantes**

Variáveis que valores não podem ser alterados.

//definindo variáveis constantes

    define('BD\_URL', 'endereco\_bd\_dev'); //essa função precisa ser passada com dois valores(nomedavariavel, valordavariavel)

    //usar caracteres maiúsculos é uma boa prática

echo BD\_URL . '<br/>';

**If/else - Introdução**

É uma estrutura de controle;

Nos possibilita com base em uma determinada condição, definir qual fluxo a aplicação vai tomar;

Através dele que podem ser feitas verificações para criação de fluxos dentro das nossas aplicações;

Verificação que atendam a todos os tipos de possibilidades, pelo menos, todas que puderem ser previstas.

**Operadores de comparação (condicionais)**

Servem para formar as expressões condicionais para o comando if.

***Operadores:***

Igual (**==**);

Idêntico (**===**) – verifica valor e tipo;

Diferente (**!=**);

Diferente (**<>**);

Não idêntico (**!==**) – verifica se valores são diferentes e de **tipos diferentes;**

Menor (**<**);

Maior (**>**);

Menor igual (**<=**);

Maior ou igual (**>=**).

if (1 >= 2) {

        echo 'Verdadeiro';

    } else {

        echo 'Falso';

    }

    //identico e nao identico verifica valores e tipos

***Operadores lógicos***

Servem para tornar comparação de condições mais complexas, maiores.

***Operadores:***

E (**AND** ou **&&**)- true se todas as operações são verdadeiras;

OU (**OR** ou **||**)- true se pelo menos uma operação é verdadeira;

Xor (**XOR**)– true se apenas uma das operações é verdadeira;

Negação (**!**)– inverter resultado da expressão.

//Operadores lógicos

    //E: && ou AND -> retorna verdadeiro se todas as expressões forem verdadeiros

    //f e f = f

    //f e v = f - v e f = f

    //v e v = v

    //OU: || ou OR -> retorna verdadeiro se pelo menos uma das expressões é verdadeiras

    //f ou f = f

    //v ou f = v - f ou v = v

    //v ou v = v

    //XOR: XOR -> retorna verdadeiro se apenas uma das expressões for verdadeira

    //f xor f = f - v xor v = f

    //v xor f = v - f xor v = v

    //! - inverte o resultado informado pela expressão

    //v xor v = !(f) = v

    //v xor f = !(v) = f

    //v e v e f = !(f) = v

    //estabelecer precedência - interpretador do php lê primeiro o que há entre parenteses

    //(v e v) = v ou f = v

***Praticando If/else e operadores***

 <?php

$usuariopossui\_cartaoloja = true;

    $valor\_compra = 225;

    $valor\_frete = 50;

    $recebeu\_desconto\_frete = false;

    if ($usuariopossui\_cartaoloja && $valor\_compra >= 100) {

        $valor\_frete = 0;

        $recebeu\_desconto\_frete = true;

    }

    ?>

<h1>Detalhes do pedido</h1>

    <p>Possui cartão da loja?

        <?php

        if ($usuariopossui\_cartaoloja) {

            echo 'SIM';

        } else {

            echo 'NÃO';

        }

        ?>

    </p>

***Condições encadeadas***

- Tem mais de duas condições if – else if

$recebeu\_desconto\_frete = true;

if ($usuariopossui\_cartaoloja && $valor\_compra >= 400) {

        $valor\_frete = 0;

    } else if ($usuariopossui\_cartaoloja && $valor\_compra >= 300) {

        $valor\_frete = 10;

    } else if ($usuariopossui\_cartaoloja && $valor\_compra >= 100) {

        $valor\_frete = 25;

    } else {

        $recebeu\_desconto\_frete = false;

    }

**Operador Ternário**

$valor\_frete\_aux = $usuariopossui\_cartaoloja && $valor\_compra >= 400 ? 0 : ($usuariopossui\_cartaoloja >= 300 ? 10 : ($usuariopossui\_cartaoloja && $valor\_compra >= 100 ? 25 : $valor\_frete));

    $recebeu\_desconto\_frete = $valor\_frete != $valor\_frete\_aux ? true : false;

    $valor\_frete = $valor\_frete\_aux;

**Switch**

Parece com o if mas é mais a sintaxe é mais simples.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*explicação do método:\*/

        var opcao = 2

        switch(opcao){ //opcao  -->variavel ou valor pré-definido

            case 1:/\* -- > função - comparar o valor encaminhado com o switch com o valor do case

                     -- > comparação feita utilizando o operador idêntico\*/

                //trecho de código que será executado

                break//precisa ser utilizado no final de cada

            case 2:

                //trecho de código que será executado

                break

            default://funciona como um else

                //trecho de código que será executado

                break

        }/\* -->nesse exemplo, o case que será disparado, será o 'case 2', por conta da comparação que será feita com a variável(nesse caso)'opcao'

        /

Default é opcional;

Precisa haver um break ao final de toda função;

Break é importante para interromper funções de um determinado case. Pois programa roda até achar um break;

Se nenhuma condição for satisfeita, loop entra no default;

Os cases podem ser de qualquer tipo.

**Casting de tipos com (int), (bool), (float) e (string)**

Float e double são as mesmas coisas;

Função ***gettype*** ajuda no debug;

Quando é convertida uma fração do tipo float para int, preserva apenas a parte inteira

//gettype() => retorna o tipo da variável -> importante nesse momento para debug

    //$valor = 10;

    //$valor2 = (float) $valor;//float, double

    //$valor2 = (string) $valor;

    $valor = true;

    //$valor2=(int)$valor;//preserva apenas parte inteira

    //$valor2=(string)$valor;//preserva apenas parte inteira

    $valor2 = (string) $valor; //boolean ou bool -  boolean para int retorna um ou zero

    //fato da variável representar um valor válido retorna true na conversão para bool- caso false não retorna nada

    $valor3=(int)$valor2;

**Operações aritméticas**

Operadores matemáticos que nos permitem efetuar cálculos.

Operadores

Adição (**+**)

Subtração (**-**)

Multiplicação (**\***)

Divisão (**/**)

Módulo (**%**)

Módulo só retorna número inteiro.

echo "A módulo entre $num1 e $num2, é: ".($num1%$num2);

        echo'<br/>'

//=

    //+,-,\*,/,%

    $x = 10;

    $x = $x + 5;

    $x%=50;

    echo $x;

**Operadores de incremento e decremento**

É acrescentado(incremento) ou retirado(decremento) uma unidade do valor atual.

Pré-incremento (**++$a**)

$a = 7;

    echo "O valor contido em a é $a <br/>";

    echo 'O valor contido em a previamente do incremento, é '. ++$a. '<br/>'; //incremento é feito antes da interpretação da instrução//result:8

    echo "O valor atualizado é $a"; //result:8

Pós- incremento (**$a++**)

$a = 7;

    echo "O valor contido em a é $a <br/>";

    echo 'O valor contido em a após o incremento, é '. $a++. '<br/>'; //neste momento o valor ainda é o inicial, pois o incremento é feito após a interpretação da instrução

    echo "O valor atualizado é $a"; //valor=8

Pré- decremento (**--$a**)

$a = 7;

    echo "O valor contido em a é $a <br/>";

    echo 'O valor contido em a previamente do decremento, é '. --$a. '<br/>'; //decremento é feito antes da interpretação da instrução//result:6

    echo "O valor atualizado é $a"; //result:6

Pós-decremento (**$a--**)

$a = 7;

    echo "O valor contido em a é $a <br/>";

    echo 'O valor contido em a previamente do decremento, é '. --$a. '<br/>'; //decremento é feito antes da interpretação da instrução//result:6

    echo "O valor atualizado é $a"; //result:6

**Funções**

***Introdução***

Encapsulamento de um bloco de códigos com um objetivo definido;

Onde se escreve uma lógica que se torna reutilizável; gastando assim, menos linhas de código;

Caso tenha que ser feita alguma mudança na função, basta mudar o local onde a função foi criada e todos os pontos onde ela está sendo chamada, contam com essa atualização.

Declaração de uma função:

//void - pois não tem retorno

function exibirBoasVindas()

    {

        echo = “Bem vindo ao curso de PHP”;

    }

//return(com retorno)

function calcularAreaTerreno($largura, $comprimento)

    {

        $area = $largura \* $comprimento;

        return $area;

    }

echo calcularAreaTerreno(2,3)//como está sendo feita uma multiplicação, valores precisam ser passados obrigatoriamente

Palavra function;

Nome da função. Padrão:

(1ºverbo, 2ºsubstantivo),

Começa com letra minúscula e se tiver mais palavras começa com letra maiúscula;

Parâmetros recebidos da função (entre parênteses) - se tiver;

Entre as chaves fica o bloco de código da função.

Parâmetros funcionam como entrada de dados da função;

Funções são como subprogramas;

Podem ser de dois tipos diferentes: void (quando há apenas o processamento de alguma lógica) ou do tipo com retorno (além de processar alguma lógica, retorna um valor para script que efetuou a chamada da função).

$resultado = calcularAreaTerreno(20, 10);

    echo $resultado

Maneira de exibir valor retornado.