UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

Programação Web e SGC

Sintaxe php

Docentes: Ruben Manhiça

Maputo, 15 de maio de 2024

Conteúdo da Aula

1. Sintaxe php
2. Variáveis em php
3. Tipos de dados em php

Sintaxe

- O código PHP fica embutido entre as tags HTML. O interpretador do PHP identifica quando inicia e termina uma instrução ou bloco de instruções em PHP quando são usadas as seguintes tags:

```
<?php
   Instruções
?>
<script language="php">
   instruções
</script>
```

Habilitados por padrão

```
<?
   instruções
?>
```

Forma abreviada da primeira opção, para funcionar deve-se habilitar no arquivo php.ini a opção *short_open_tag* para *on*

```
<%
   instruções
%>
```

Esta forma foi criada para desenvolvedores acostumados com a linguagem asp, para funcionar deve-se habilitar no arquivo php.ini a opção *asp_tags* para *on*

Sintaxe

- Ao escrever um código em php, da mesma forma da linguagem C/Java, devemos colocar um ponto e vírgula (*;*) separando cada instrução.

```
<?php
    echo “meu código em php”;
?>
```

- No caso de apenas uma instrução ou da última instrução, antes do fechamento da tag PHP, o uso do ponto e vírgula (;) não é obrigatório.

```
<?php
    echo “instrução 1”;
    echo “instrução 2”
?>
```

- Por questão de padrões sempre utilizamos o ponto e vírgula ao fim de cada instrução.

Sintaxe

- Strings em PHP
  - Quando escrevemos uma string, podemos utilizar aspas (“ ”) ou apóstrofos (‘ ’) para delimitá-la.

```
<?php
  echo “exemplo string com aspas”;

  echo ‘exemplo string com apóstrofos’ ;
?>
```

  - Quando precisamos escrever uma aspa ou um apóstrofo dentro de uma string, precisamos “escapá-la”, utilizando uma contra-barra (\ )

```
<?php
  echo “exemplo aspas \“ dentro da string com aspas”;

  echo ‘exemplo apóstrofo \’ dentro da string com apóstrofos’ ;
?>
```

Sintaxe

- Strings em PHP
  - A diferença entre o uso de aspas e apóstrofo é que quando utilizamos aspas, o PHP consegue interpretar caracteres de escape especiais e ainda, faz a interpretação de variáveis.

```
<?php
  echo “exemplo string com \r retorno de carro”;

  echo “exemplo string fim de linha \n”;

  echo “texto”;

  $nome = “João”;
  echo “Seu nome é $nome”;

?>
```

Sintaxe - Comentários

- O PHP suporta 2 estilos de comentário.
  - Comentários de uma linha utilizando # ou //:
  - Comentários de mais de uma linha utilizando /* .. */

```
<?php
   # Comentário 1
  echo “exemplo de comentário”;

   // Comentário 2
  echo “exemplo de comentário”; // Comentário 3

  /* Comentário com mais de
  uma linha */
  echo “exemplo de comentário”;
?>
```

Sintaxe - Comentários

- Nossa primeira página em PHP:

```
<html>
    <head>
      <title>Primeira página em PHP</title>
    </head>
    <body>
      <?php
                echo “Este é meu primeiro script em PHP”;
                echo “<p>Posso escrever tags HTML”;
                echo “dentro dos textos em PHP</p>”;
      ?>
    </body>
</html>
```

Sintaxe - Declaração de Variáveis

- O uso de variáveis no PHP requer algumas regras:
  - Toda variável começa pelo caractere cifrão ($); Após o cifrão ($) deve-se colocar uma string que deve começar por uma letra ou pelo caractere underline (_);
  - Após esses caracteres só podem ser inseridos letras ou números.
- No PHP as variáveis são *case sensitive*, ou seja, $var, $Var e $VAR são variáveis diferentes e podem possuir valores diferentes.
- O PHP possui conversão automática de tipo, chamada também de coerção de tipo automática. Assim, a definição de tipo explicita não é necessária na declaração de variáveis.

Sintaxe - Declaração de Variáveis

- O tipo de uma variável é determinado pelo contexto em que a variável é utilizada. Isto significa que, se você assimila um valor string para a variável $var, $var se torna uma string. Se você então assimila um valor inteiro para $var, ela se torna um inteiro.

- Exemplo de variáveis PHP:

```
$variavel = “valor”;
$_variavel = 0;
$nome = “João”
$cidade = “Maputo”
$mes1 = “Janeiro”
```

Sintaxe - Declaração de Variáveis

- Type Casting – Coerção de tipo explícita
  - A coerção de tipos no PHP funciona como no C: o nome de um tipo desejado é escrito entre parênteses antes da variável em que se deseja a coerção.

- As coerções permitidas são:
  - (int), (integer) - coerção para inteiro
  - (bool), (boolean) - coerção para booleano
  - (float), (double), (real) - coerção para número de ponto flutuante
  - (string) - coerção para string
  - (array) - coerção para array
  - (object) - coerção para objeto

Sintaxe - Constantes

- No PHP, a declaração de constantes também possui algumas regras:
  - Constantes devem ser declaradas em maiúsculas.
  - Constantes não podem ter um sinal de cifrão ($) antes delas;
  - Constantes só podem ser definidas utilizando a função define();
  - Constantes podem ser definidas e acessadas de qualquer lugar sem que as regras de escopo de variáveis sejam aplicadas;
  - Constantes não podem ser redefinidas ou eliminadas depois que elas são criadas;
  - Constantes só podem conter valores escalares.
- Exemplo de declaração de constantes:

```
<?php
    define(“VARIAVEL”,"valor");
    echo VARIAVEL;
?>
```

Sintaxe - Tipos de Dados

- O PHP trabalha com os tipos de dados:
  - Boolean
  - Integer
  - Float
  - String
  - Array
  - Object
  - Resource
  - Null

Sintaxe - Tipos de Dados

- Booleanos
  - Para especificar um literal booleano, use as palavras chave TRUE ou FALSE. Ambas são insensitivas ao caso.
- Inteiros
  - Inteiros podem ser especificados em notação decimal (base 10), hexadecimal (base 16) ou octal (base 8), opcionalmente precedido de sinal (- ou +). O tamanho de um inteiro é dependente de plataforma, sendo um numero aproximado a 2 bilhões o valor mais comum (número de 32 bits com sinal).
  - Overflow de inteiros
    - Se você especifica um número além dos limites do tipo inteiro, ele será interpretado como um ponto flutuante.

Sintaxe - Tipos de Dados

- Números de pontos flutuantes
  - O tamanho de um número de ponto flutuante é dependente de plataforma, sendo o máximo com uma precisão de 14 decimais digitais.
- Strings
  - Uma string é uma série de caracteres. No PHP, um caracter é o mesmo que um byte, ou seja, há exatamente 256 caracteres diferentes possíveis. Não há nenhum problema se as strings se tornarem muito grandes. Não há nenhum limite para o tamanho de strings imposta pelo PHP, então não há razão para se preocupar com strings longas.
- Arrays
  - Um array no PHP é atualmente um mapa ordenado. Um mapa é um tipo que relaciona valores para chaves. Este tipo é otimizado de várias maneiras, então você pode usá-lo como um array real, ou uma lista (vetor), hashtable (que é uma implementação de mapa), dicionário, coleção, pilha, fila e provavelmente mais. Como você pode ter outro array PHP como um valor, você pode facilmente simular árvores.

Arrays

- São estruturas de dados que podem armazenar múltiplos valores

Exemplo:

```
$cores= array('vermelho', 'verde', 'azul');

$cor = $cores[1]; // retorna "verde" (2º elemento do array)

$cores[1]='amarelo'; // atribui novo valor (ao 2º elemento do array)
```

Arrays

- Podem ser definidos como mapeamentos **ou** vetores indexados

Exemplo:

```
<?php
  $cores[0]="Red";
  $cores[1]="Green";
  $cores[2]="Blue";
?>
```

Arrays - tipos de índices

- **Ordenado** → baseado em número (começa no 0) (indexada numericamente)
- **Associativo** → formado por caracteres alfanuméricos (indexada por nome)

Arrays

```
// cria e inicializa um array (indexada
   numericamente)
$cores = array("Red", "Green", "Blue");
```

ou

```
// cria e inicializa um array usando índices
   (explicitamente)
$cores = array(0=>'Red', 1=> "Green", 2=>"Blue");
```

ou

```
// cria e inicializa um array usando índices
   (numéricos)
$cores[]="Red";
$cores[]="Green";
$cores[]="Blue";
```

Arrays Associativos

São conjunto ordenados de **chaves** e **valores**, onde cada valor é acessado através de uma chave associada.

Exemplo:

```
$provincia_e_capital = array (
  'MP' => 'Matola',
  'SF' => 'Beira',
  'TT' => 'Tete',
  'MN' => 'Manica'
);
```

Arrays Associativos

```
<?php
 $cor['red']=0;
 $cor['green']=255;
 $cor['blue']=0;
?>
```

ou

```
<?php
  $cores=array('red'=>0, 'green'=>1, 'blue'=>2);
?>
```

Arrays (3)

- Usamos a função unset() para destruir todo o array.

Funções de array

- array_pop($array) → retira e retorna o último elemento do array
- array_push ($array,$var) → insere um ou mais elementos no fim de um array
- array_shift ($array) → retira e retorna o primeiro elemento de um array
- array_unshift ($a,$val) → insere um novo elemento no inicio de um array
- array_rand ($array) → retorna um ou mais elementos do array
- array_reverse ($array) → retorna um array com ordem inversa
- array_keys ($array) → retorna as chaves de um array
- array_values ($array) → retorna os valores de um array
- sizeof () → retorna o número de elementos do array
- count () → retorna a quantidade de elementos de um array

Sintaxe - Tipos de Dados

- Objetos
  - Utilizado na orientação a objetos
- Resource
  - Resource (Recurso) é uma variável especial, mantendo uma referência de recurso externo. Resources são criados e utilizados por funções especiais. Por exemplo, quando criamos uma conexão com uma base de dados, o PHP retorna uma variável, a qual possui o status da conexão dentro do sistema.
- NULL
  - O valor especial NULL representa que a variável não tem valor.

Sintaxe - Operadores

- Operadores aritméticos:
  - $x + $y – Adição;
  - $x - $y – Subtração;
  - $x / $y – Divisão;
  - $x * $y – Multiplicação;
  - $x % $y – Módulo;

- Operadores de Atribuição:
  - O operador de atribuição é o igual (=)
  - $x = ($y = 5) + 1; $y recebe o valor 5 e $ x recebe o valor da soma entre $y e 1. $x recebe o valor 6;

Sintaxe - Operadores

- Operadores de comparação:
  - $x == $y – igual;
  - $x === $y – idêntico;
  - $x != $y – diferente;
  - $x > $y – maior que;
  - $x < $y – menor que;
  - $x >= $y – maior ou igual a;
  - $x <= $y – menor ou igual a;

- Operadores de controle de erro:
  - O PHP suporta um operador de controle de erro: o caractere arroba (@). Quando ele precede uma expressão em PHP, qualquer mensagem de erro que possa ser gerada por aquela expressão será ignorada.
    - @gettype($variavel);

Sintaxe - Operadores

- Operadores de incremento/decremento:
  - ++$x – pré-incremento;
  - $x++ – pós-incremento;
  - --$x – pré-decremento;
  - $x-- – pós decremento;

- Operadores lógicos:
  - $x and $y – verdadeiro se $x e $y forem verdadeiros.
  - $x or $y – verdadeiro se $x ou $y forem verdadeiros.
  - $x xor $y – verdadeiro se $x ou $y forem verdadeiros, mas não ambos.
  - !$x – verdadeiro se $x for falso.
  - $x && $y – verdadeiro se $x e $y forem verdadeiros.
  - $x || $y – verdadeiro se $x ou $y forem verdadeiros.

Sintaxe - Operadores

- Operadores de texto (concatenação)
  - Para concatenar textos utilizamos o operador ponto (.)

```
$ a = “a”; $b = “b”;
$c = $a.$b; //$c = “ab”;
```

  - Para atribuir e concatenar textos utilizamos o operador ponto e igual (.=)

```
$c = “ab”; $c .= “c”; //$c igual a “abc”
```

- Operadores de array:
  - $x + $y – união entre $x e $y;
  - $x == $y – igualdade – se possui os mesmos elementos;
  - $x === $y – identidade – se possui os mesmos elementos na mesma ordem;
  - $x != $y e $x <> $y – diferença;
  - $x !== $y – não possui a mesma identidade;

Sintaxe - Estruturas de decisão

- if...else... elseif
  - No PHP, assim como nas outras linguagens baseadas em C, os valores a serem comparados dentro deste tipo de estrutura devem estar entre parenteses.

```
if ($x == $y)
        echo “x é igual y”;
```

  - Quando utilizamos mais de uma instrução dentro deste tipo de estrutura, é necessário o uso de chaves para determinar o início e fim da estrutura.

```
if ($x > $y) {
        echo “x é maior que y;
        $x = $y;
}
```

Sintaxe - Estruturas de decisão

- if...else...elseif
  - Uso do else

```
if ($x == $y){
        echo “x é igual y”;
}else{
        echo “x é diferente de y”;
}
```

  - Uso do elseif

```
if ($x == $y){
        echo “x é igual y”;
}elseif ($x > $y){
        echo “x é maior que y”;
}else{
        echo “x é menor que y”;
}
```

Sintaxe - Estruturas de decisão

- switch – estrutura que substitui estruturas if aninhadas

```
switch($i){
   case 0:
            echo “i é igual a 0”;
            break;
   case 1:
            echo “i é igual a 1”;
            break;
   case 2:
            echo “i é igual a 2”;
            break;
   default:
            echo “i é diferente de 0, 1 e 2”;
}
```

Sintaxe - Estruturas de laço

- Estrutura while – executa um bloco de instruções enquanto uma condição for satisfeita. O bloco de instruções é executado somente após a verificação da condição.

```
<?php
    $i = 0;
    while ($i < 10){
                    echo $i;
                    $i++;
    }
?>
```

- Estrutura do ... while – parecida com a estrutura while. A diferença é que aqui o bloco de instruções é executado antes da verificação da condição.

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

Sintaxe - Estruturas de laço

- Estrutura for – utilizada quando precisamos repetir um bloco de instruções por um determinado número de vezes.

```
<?php
    for ($i = 0; $i <= 10; $i++){
        echo $i;
    }
?>
```

TPC

1. Escreva um programa em PHP e Html que receba um valor digitado pelo usuário e verifique se esse valor é positivo, negativo ou igual a zero. Imprima na tela: "Valor Positivo", "Valor Negativo", "Igual a Zero";
2. Efetue um algoritmo em PHP e Html que receba um valor qualquer e imprima os valores de 0 até o valor recebido.

   exemplo:

   Valor recebido = 9

   Impressão do programa – 0 1 2 3 4 5 6 7 8 9

FIM!!!

Duvidas e Questões?