PHP básico

Sintaxe Básica

Tipos, variáveis e operadores

Estruturas de controle e de fluxo

Material Didático organizado pela Profª. Aline de Campos

Padrões

Arquivos comuns	**.php**
Arquivos html	**.html**
Arquivos incluídos	**/includes e ou extensão .inc.php**
Arquivos de classes	**/class e ou extensão.class.php**
Arquivos Javascript	**/javascript e extensão .js**
Arquivos de imagens	**/imagens**

Dicas...

Antes de começarmos, algumas dicas...

1) Comente seu código com frases que realmente façam sentido.
2) Utilize identação! Ela ajuda você e outras pessoas na visualização do código.
3) Utilize nomes descritivos para variáveis e funções.
4) Decida-se entre usar português ou inglês, não misture os dois.
5) Organize bem as chaves, colchetes e parênteses, eles podem causar uma grande confusão mesmo em um código simples.
6) É interessante inserir comentários de data e nome de quem realizou as últimas alterações no código quando se está trabalhando em grupo.
7) Quando for realizar uma alteração grande e está inseguro quanto ao resultado, lembre-se "*O backup é nosso pastor e nada nos faltará!*".
8) Na ocorrência de erros, preste atenção ao que o código de erro lhe diz, em geral são erros simples de corrigir originados por pequenos descuidos.

Delimitação do código PHP

Para o servidor executar o interpretador PHP é necessário que o código esteja delimitado com as tags <?php e ?>.

```
<?php
      echo "Primeira forma de delimitação do PHP";
?>
```

Esta segunda maneira necessita da habilitação da opção short-tags na configuração do PHP (php.ini).

```
<? echo "Segunda forma de delimitação do PHP"; ?>
```

Separador e Case Sensitive

Separador de instruções em PHP é ponto-e-vírgula.

```
<?php echo "Separando instruções"; ?>
```

Case Sensitive, quer dizer que o interpretador diferencia variáveis maiúsculas e minúsculas.

```
<?php
        $variavel = "valor";
        $VARIAVEL = "valor";
?>
```

Comentários em PHP

Para apenas uma linha utiliza-se duas barras //.

```
<?php
  // Inserindo comentários
  echo "Exemplo de Comentários"; // echo
?>
```

Mais de uma linha deve-se usar os caracteres /* para o início e os caracteres */ para o final do comentário.

```
<?php
    /* Comentário de mais
    de uma linha */
?>
```

PHP e HTML juntos

Páginas em PHP podem conter marcações de HTML. Parte dinâmica fica com o PHP o estática com HTML, auxiliando na exibição dos dados no navegador.

```
<html>
  <head><title>PHP e HTML</title></head>
  <body>
    <?php $usuario = "João"; ?>
    <p> Bem vindo (a) <?php echo $usuario; ?></p>
  </body>
</html>
```

Constantes

Definidas no início do script e não mudam seu valor.
As constantes pré-definidas são:

__FILE__	nome do arquivo.
__LINE__	número da linha do arquivo.
PHP_VERSION	versão do analisador PHP em uso.
PHP_OS	sistema operacional
TRUE	valor verdadeiro.
FALSE	valor falso.
E_ERROR	erro, com execução interrompida.
E_WARNING	mensagem, sem execução interrompida.
E_PARSE	erro sintaxe, execução interrompida.
E_NOTICE	notifica, sem execução interrompida.

Constantes

Definição de constantes com a função define().

```php
<?php
  // "João da Silva" para a constante NOME
  define("NOME","João da Silva");

  // Recurso de concatenação (.)
  echo "O nome é ".NOME;
?>
```

Include e Require

Tanto a função Include quanto a função Require, inclui e analisa o arquivo informado. A diferença entre eles é como os erros são tratados. No Require é produzido um erro do tipo Fatal Error e a execução do Script é finalizada. Já o include produz em erro Warning e continua a execução.

```php
<?php
  include "conexao.php";
  require "conexao.php";
?>
```

Variáveis

Variáveis armazenam dados para serem usados em qualquer ponto. Devem começar sempre com o caracter $ seguido de um identificador que nunca poderá ser um número embora números possam aparecer em outras posições do identificador.

```php
<?php
    // ERRADO
    $10teste = "Inválido";

    // CERTO
    $teste10 = "Válido";
?>
```

Variáveis

Tipagem dinâmica: não é necessário declarar variáveis.
Uma variável com tipos diferentes durante a execução.

Inteiro: variável conterá valores Integer ou Long.

```
$numero = 1234;   // positivo na base decimal
$numero = -234;  // negativo na base decimal
$numero = 0234;  // base octal - 0 equivale a 156 decimal
$numero = 0x34;  // base hexadecimal(simbolizado pelo 0x) –
                 // equivale a 52 decimal.
```

Ponto Flutuante: variável conterá valores Float ou Double.

```
$numero = 1.234;
$numero = 23e4;  // equivale a 230.000
```

Variáveis

Strings são representadas de duas formas:

1) com aspas simples: o valor será exatamente o texto contido entre as aspas
2) com aspas duplas: qualquer variável ou caractere de escape será expandido antes de ser atribuído. Ou seja utilizado aspas duplas variáveis e caracteres de escape serão reconhecidos.

```php
<?php
    $nome = "João";
    $texto = 'Oi\n***$nome***';
    echo "$texto";
    // Saída será: Oi\n***$nome***.

    $texto = "Oi\n***$nome***";
    echo "$texto";
    /* Saída será: Oi
                   ***João***  */
?>
```

Caracteres de escape

Sintaxe	Significado
\n	Nova linha
\r	Retorno de carro (semelhante a \n)
\t	Tabulação horizontal
\\	A própria barra (\ )
\$	O símbolo $
\’	Aspa simples
\”	Aspa dupla

Estruturas de controle - IF

Uma condição é avaliada retornando um valor lógico, verdadeiro ou falso, onde dependendo do resultado uma instrução é executada. A sintaxe é a seguinte:

SINTAXE

```
if(condição)
{ bloco de código }
elseif (condição)
{ bloco de código }
else
{ bloco de código }
```

Estruturas de controle - IF

```php
<?php
 if($nota >= 7) {
    echo "Você foi aprovado!";
 } elseif(($nota >= 3) && ($nota < 7)){
    echo "Você está em exame.";
 } else {
    echo "Você foi reprovado.";
 }
 ?>
```

Estruturas de controle - SWITCH

Quando uma variável pode receber diversos valores.

SINTAXE

```
switch(operador) {
   case valor1: <instruções> break;
   case valor2: <instruções> break;
   case valor3: <instruções> break;
   default: <instruções> break;
}
```

Estruturas de controle - SWITCH

```php
<?php
  switch($texto) {
     case 's':
     echo "A resposta é SIM.";
     break;
     case 'n':
     echo "A resposta é NÃO.";
     break;
     default:
     echo "Opção inválida.";
     break;
  }
?>
```

Estruturas de controle - WHILE

Laço de repetição que executa instruções enquanto expressão retornar um valor verdadeiro.

SINTAXE

```
while(expressão)
{
   <instruções>
}
```

Estruturas de controle - WHILE

```php
<?php
 $i = 1;
 while($i < 10)
 {
    echo "O valor é $i <br/>";
    $i++;
 }
 ?>
```

Estruturas de controle – DO WHILE

O do while executa primeiro o bloco de instruções e depois avalia a expressão. Indicado em situações que o laço precisa ser executado obrigatoriamente pelo menos uma vez.

SINTAXE

```
do {
   <instruções>
} while(expressão);
```

Estruturas de controle – DO WHILE

```php
<?php
 $i = 1;
 do
 {
     echo "O valor é $i <br/>";
     $i++;
 } while($i < 10);
?>
```

Estruturas de controle – FOR

Laço de repetição indicado quando queremos executar instruções um determinado número de vezes.

SINTAXE

```
for(inicial; condição; operador)
{
   <instruções>
}
```

Estruturas de controle – FOR

```php
<?php
    for($i=0; $i<10; $i++) {
        echo "O valor é $i <br/>";
    }
?>
```

Operadores - Aritméticos

Sintaxe	Significado
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto de Divisão
-op	Trocar sinal
++op	Pré-incremento
--op	Pré-decremento
op++	Pós-incremento
op--	Pós-decremento

Operadores - Binários

Sintaxe	Significado
&	“e” lógico
|	“ou” lógico
^	ou exclusivo
~	não (inversão)
<<	desloca bits à esquerda
>>	desloca bits à direita

Operadores - Comparação

Sintaxe	Significado
==	igual a
!=	diferente de
<	menor que
>	maior que
<=	menor ou igual a
>=	maior ou igual a

Operadores - Atribuição

Sintaxe	Significado
=	atribuição simples
+=	atribuição com adição
-=	atribuição com subtração
*=	atribuição com multiplicação
/=	atribuição com divisão
%=	atribuição com módulo
.=	atribuição com concatenação

Operadores – Lógico

Sintaxe	Significado
And	“e” lógico
Or	“ou” lógico
Xor	ou exclusivo
!	não (inversão)
&&	“e” lógico
||	“ou” lógico

Precedência de operadores

Precedência	Associatividade	Operadores
01	esquerda	,
02	esquerda	or
03	esquerda	xor
04	esquerda	and
05	direita	print
06	esquerda	= += -= *= /= .= %= &= != ~= <<= >>=
07	esquerda	? :
08	esquerda	||
09	esquerda	&&
10	esquerda	|
11	esquerda	^
12	esquerda	&
13	não associa	== !=
14	não associa	< <= > >=
15	esquerda	<< >>
16	esquerda	+ - .
17	esquerda	* / %
18	direita	! ~ ++ -- (int) (double) (string) (array) (object) @
19	direita	[
20	não associa	new