Curso de Operador de Tecnologia da Informação e Comunicações

Desenvolvimento Web

PHP - Programação

Objetivo de Aprendizado

Entender quais os tipo de dados suportados pelo PHP e quais os blocos de comandos mais utilizados em programação

A linguagem PHP é um das mais utilizadas no mundo devido a sua versatilidade e também por causa da facilidade de aprendizado. Existem várias ferramentas que podem ser utilizada com o PHP relacionadas a frameworks que podem ajudar no desenvolvimento rápido e seguro.

Sintaxe básica em PHP

Todo arquivo em PHP possui a extensão .php. O PHP interpreta o código escrito dentro das tags de abertura e fechamento, e o resto do código é ignorado pelo interpretador do PHP. Existem diferentes pares de tags de abertura e fechamento que podem ser usadas no código PHP Achour et al. [2014], são elas:

```
<?php
   echo'Primeiro par de tags PHP';
?>
```

O código em PHP também pode ser embutido no documento HTML, como no exemplo abaixo:

```
<html>
<body>
     <h1> Exemplo de PHP </h1>
<?php
   $a = 10;
   echo $a;
?>
</body>
</html>
```

SEPARAÇÃO DE INSTRUÇÕES

Na linguagem PHP, assim como em C ou Perl, todo comando deve terminar com ponto-e-vírgula (;). Em PHP, o último comando antes da tag de fechamento não precisa ter um ponto-e-vírgula para finalizar o comando.

```
<?php
  echo'Exemplo';
  $a=$b+ 3;
?>


<?php
  echo 'Exemplo sem ponto-e-vírgula'
?>
```

COMENTÁRIOS EM CÓDIGOS

O PHP suporta comentários no estilo C e Shell Script. O comentário de linha pode começa com//ou#e termina no final da linha ou com a tag de fechamento(?≤), e o comentário de múltiplas linhas começa com (/∗) e termina com(∗/).

```
<?php
  echo'Exemplos de comentários';
  // Estilo de comentário de uma linha


  /*Estilo de comentário para
     múltiplas linhas*/


  # Estilo de comentário de uma linha com em ShellScript

  # echo 'Exemplo de comentário';?> A partir não é mais
  comentário</p>
```

VARIÁVEIS

O nome da variável sempre começa com o caracter **'$'** seguida do nome da variável. O primeiro caracter do nome pode ser uma **letra** ou o caracter **'_'** e o restante do nome pode conter os caracteres alfanuméricos e o caracter '_' (A-z,0-9,_).

```
<?php

$variavel;

$_Variavel123;

$abc;
$ABC;
?>
```

O PHP é **case sensitive**, isto é, letras maiúsculas e minúsculas são diferentes. No exemplo acima, a variável $abc não é a mesma variável que a variável $ABC.

Ao contrário das outras linguagens, em PHP **não é necessário declarar as variáveis** que serão usadas, tampouco definir seu **tipo**. Por exemplo, ao inicializar a variável $a com o valor 3, o PHP reconhecerá que a variável $a é do tipo inteiro. $a = 3;

TIPOS DE DADOS

O PHP suporta os seguintes tipos de variáveis: inteiros, pontos flutuantes, strings, booleanos, array e objetos.

Booleano

O tipo booleano expressa um valor lógico que pode ser verdadeiro(TRUE) ou falso(FALSE).

Dados	*Descrição*
TRUE	*Verdadeiro*
FALSE	*Falso*

Exemplo de variáveis booleanas.

$var = TRUE;

$var2 = FALSE;

Inteiros

Uma variável inteira pode ser especificada em notação decimal, octal ou em hexadecimal.

Exemplos

Dados	*Descrição*
$\$numero1 = 210$	Número inteiro na base decimal
$\$numero2 = -5$	Número inteiro negativo
$\$numero3 = 0123$	Número inteiro na base octal (83 na base decimal)
$\$numero4 = 0x1A$	Número inteiro na base hexadecimal (26 na base decimal)

Ponto flutuante

Um ponto flutuante pode ser especificado utilizando uma das seguintes sintaxes:

Dados	*Descrição*
$\$numero1 = .15$	Número real com duas casas decimais (0.15)
$\$numero2 = 4e + 6$	Número real grande 4000000
$\$numero3 = 2e - 5$	$2/100000$

Strings

String é um tipo de dados formado por uma sequência de caracteres alfanuméricos. No PHP é permitido declarar uma string com aspas duplas “ ” ou aspas simples ‘’

```
<?php

$str = ‘IFSP’;  //o conteúdo da string é conservado

$str = "IFSP";  //O PHP irá interpretar mais sequências de escape


?>
```

Sequências de escape

Sequência	*Descrição*
$\backslash n'$	Nova linha
$\backslash t'$	Tabulação horizontal
$\backslash''$	Aspas duplas
$\backslash\$$	o caracter $
$[a-zA-Z]*$	Expressão regular para qualquer string com letra maiúscula ou minúscula.

Arrays

Um variável do tipo array pode ser composto por chaves dos tipos inteiro e string. Exemplos de inicialização de arrays.

```
$vetor=array(1=>false, 2=>false, 3 =>true);
```

A variável ***vetor*** no exemplo acima possui as chaves 1,2 e 3 e valores booleanos false, false e true.

```
$vetor=array(1, 2, 3);
$vetor2=array(10 => 3, 11 => 6, 12 => 9,"a"=>100);
```

OPERADORES

O PHP oferece uma ampla variedade de operadores, entre binários e unários. A seguir serão apresentados os operadores básicos.

Operadores aritméticos

Operador	*Nome*	*Exemplo*
+	Adição	$*a*+$*b*
−	Subtração	$*a*−$*b*
−	Negação	−$*a*
*	Multiplicação	$*a**$*b*
/	Divisão	$*a*/$*b*
%	Módulo	$*a*%$*b*

Exemplo

```
<?php
    $a = 15;
    $b = 5;
    $c = $a + $b; // $c = 20
    $d = $c - $b; // $d = 15
    $e = $b * 2; // $e = 10
    $f = $c / 10; // $e = 2
    $g = $e / 3;
    // $e = 1
?>
```

Operadores de incremento e decremento

Operador	*Nome*	*Descrição*
+ +$*a*	Pré-incremento	Incrementa $a em um, e então retorna $a
− −$*a*	Pré-decremento	Decrementa $a em um, e então retorna $a
$*a*+ +	Pós-incremento	Retorna $a, e então incrementa $a em um
$*a*− −	Pós-decremento	Retorna $a, e então decrementa $a em um

Exemplos

```
<?php

$a = 10;
echo ++$a; echo $a++;

$b = 5;
echo $b--;

echo --$b;

?>
```

Operadores de string

Operador	*Nome*	*Exemplo*
=	Concatenação	$*a*=$*b*
.=	Atribuição com concatenação	$*a*.=$*b*

Exemplo

```
<?php
        $a = "IFSP";
        $b = $a . " - Guarulhos";
        $a .= " - São Paulo";
?>
```

Operadores de comparação

O operador de comparação é usada comparar dois valores.

Operador	*Nome*	*Exemplo*
==	Igual	$*a*==$*b*
! =	Diferente	$*a*!=$*b*
<>	Diferente	$*a* <>$*b*
===	Idêntico	$*a*===$*b*
! ==	Não idêntico	$*a*!==$*b*
>	Maior que	$*a*>$*b*
<	Menor que	$*a*<$*b*
>=	Maior ou igual que	$*a*>=$*b*
<=	Menor ou igual que	$*a*<=$*b*

Exemplo

```
<?php
   $a=1;
   $b="001";
   $c = ($a == $b); // true
   $d = ($a === $b); //false
   $e = 2;
   $f = ($e <= $a); //false
?>
```

Operadores lógicos

O operador de comparação é usada comparar dois valores.

Operador	*Nome*	*Exemplo*
and	E lógico	($*a and*$*b*)
&&	E lógico	($*a*&& $*b*)
or	OU lógico	($*a or*$*b*)
||	OU lógico	($\|\|$*b*)
xor	OU exclusivo	($*a xor*$*b*)
!	Não	!$*a*

OBS: XOR é o operador de OU EXCLUSIVO. Ele é utilizado quando você quer verificar a veracidade de uma expressão OU outra, exclusivamente.
Exemplo: Eu tenho 2 horas por semana para estudar ou PHP ou JAVA.
Neste caso, estamos falando em operador exclusivo. Ou um ou outro, nunca ambos. Ele retornar verdadeiro somente se um ou outro for verdade.

Exemplo

```php
<?php
  $a=1;
  $b="001";
  $c=2;


  if($a==$b)and($a<$c)
   echo "Passou pelas condições.";
?>
```

Operadores de arrays

O operador de arrays é usado para comparar dois array.

Operador	*Nome*	*Exemplo*
+	União	$*a*+$*b*
==	Igualdade	$*a*==$*b*
! =	Desigualdade	$*a*!=$*b*
<>	Desigualdade	$*a* <>$*b*
===	Idêntico	$*a*===$*b*
! ==	Não idêntico	$*a*!==$*b*

O operador de união (+) acrescenta os elementos do array da direita no array da esquerda, mas as cha-ves duplicadas não são sobrescritas. Exemplos:

```php
<?php
  $fruta1=array("a"=>"maça","b"=>"banana");
  $fruta2=array("c"=>"laranja","b"=>"melancia","a"=>"morango");


  $fruta3=$fruta1+$fruta2;
  //fruta1=array("a"=>"maça","b"=>"banana","c"=>"laranja");
  var_dump($fruta3);


  $fruta4=$fruta2+$fruta1;
 /*$fruta4 = array("c" => "laranja", "b" =>"melancia",
                  "a" => "morango");*/
  var_dump($fruta4);


  if($fruta4===$fruta2)
  echo "Iguais ";


?>
```

Estruturas de controle em PHP

Uma estrutura de controle é aquela que dado o resultado de uma expressão lógica ou relacional é possível decidir se um determinada bloco de comandos deve ou não ser executado.

Comando if

O comando ***if*** é usado para executar o bloco de código se a condição avaliada for verdadeira.

```
if(condição){
    //Bloco de comandos;
}
```

O exemplo abaixo verifica se o valor da variável $*a* é igual ao valor da variável $*b*.

```
<?php
  $a=1;
  $b=2;

  if($a==$b)
   echo"\$a = \$b";
?>
```

Pode ainda ocorrer a variação com “if...else” que é utilizado para indicar um bloco de comandos, caso a condição do if não seja satisfeita.

```
if(condição){
    //Bloco de comandos;
}else{
    //Bloco de comandos;
}
```

Exemplo

```php
<?php
  $a=1;
  $b=2;

  if($a==$b)
     echo "\$a = \$b";
  else
    echo "\$a != \$b";
?>
```

Pode ainda ocorrer a variação com “else if”. O comando **else if** é usado em situações onde precisa-se verificar **mais de uma condição**. As condições são avaliadas de cima para baixo. Assim que uma condição verdadeira for encontrada, o bloco de comandos associados à ela é executado.

```
if(condição1){
   //Bloco de comandos;
}else if(condição2){
   //Bloco de comandos;
}else{
   //Bloco de comandos;
}
```

Exemplo

```php
<?php
  $a=1;
  $b=2;

  if($a<$b)
        echo "\$a < \b";
  else if($a==$b)
        echo "\$a = \b";
   else
        echo"\$a > \$b";
?>
```

Comando switch

O comando **switch** é utilizado para fazer múltiplos testes da condição. O **switch** procura por uma coincidência entre a $variavel e outros valores associados em cada opção.

```php
<?php
   switch($variavel)
   {
   case valor1:
          //Bloco de comandos;
          break;
  case valor2:
          //Bloco de comandos;
          break;
  default:
          //Bloco de comandos. Nesse caso não igual a valor1 ou valor2;
  break;
    }
?>
```

Obs: Um tutorial mais completo desse comando pode ser encontrado em (http://php.net/manual/pt_BR/control-structures.switch.php). O comando default é executado quando nenhuma das coincidência for detectada. Exemplo do uso do switch.

```php
<?php
  $i="c";
  switch($i)
  {
    case"a":
        echo"i = a";break;
    case"b":
       echo"i =b";break;
    default:
        echo"i = c";break;
  }
?>
```

COMANDOS DE REPETIÇÃO

Um comando de repetição permite que um conjunto de instruções seja executado até que ocorrência de uma condição.

Comando while

O comando **while** testa a condição e executa um bloco de comandos enquanto a condição for verdadeira.

```
while(condição) {
    //Bloco de comandos;
}
```

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

Comando do...while

O comandodo . . . while funciona de maneira semelhante ao comandowhile, com uma pequena diferença que a condição é testada no final da execução do bloco de comandos.

```
do{
    //Bloco de comandos;
}while(condição);
```

Exemplo:

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

Obs: o teste somente é feito no final

Comando for

O comando for possui três expressões: **inicialização**, **condição** e **incremento**. A inicialização serve para inicializar o valor da variável contador. A condição do loop for é verifica a cada iteração, caso a condição for verdadeira o loop continua e caso contrário o loop é terminado. A expressão incremento/decremento aumenta ou diminui o valor da variável contador.

```
for(inicialização;condição;incremento/decremento){
    //Bloco de comandos;
}
```

Exemplo

```php
<?php
  for($i=1;$i<10;$i++){echo"
    \n i = ".$i;
  }
?>
```

Comando foreach

O comando **foreach** percorre cada item do array $variavel. Em cada item, ele armazena a chave e seu respectivo valor na variável $chave.

```
foreach($variável as $chave=>$valor){
    //Bloco de comandos;
}
```

Exemplo

```php
<?php
  $cursos=array('ADS','Automação','Matemática');

  foreach($cursos as $chave=>$nome){

  echo "<br> curso $chave = $nome";

  }
?>
```

Utilizando o **for**

```php
<?php
  $cursos=array('ADS','Automação','Matemática');

  for($i=0;$i<sizeof($cursos);$i++){

  echo "<br> curso $i = $cursos[$i]";

  }
?>
```

COMANDOS DE INTERRUPÇÕES

O PHP fornece algumas formas de interrupção antecipada de um determinado laço ou do comando ***switch***.

Comando break

O comando break pode quebrar a execução de um comando switch ou interromper a execução de qualquer comando de repetição.

Exemplo

```
<?php
  for($i=1;$i<10;$i++){
      if($x== 5){
         echo "interrompendo o for";
         break;
      }
     echo "\n i=".$i;      // imprime os valores de 1 até 4
  }
?>
```

Comando continue

O comando **continue** funciona de maneira semelhante ao comando break, com a diferença que o loop volta para o início dele.

Exemplo:

```php
<?php
  for($i=1;$i<10;$i++){
  if(($i%2) == 1)
      continue;
  echo "\n i=".$i;  // imprime os valores pares
  }
?>
```

Funções em PHP

Em PHP, além das funções embutidas, é possível criar nossas próprias funções. Uma função é um bloco de comando que pode ser usada repetidamente no programa. Quando uma função é criada esta não será executada imediatamente quando a página é carregada, mas sim quando houver uma chamada para a função.

Declaração

Para declarar uma função é preciso começar com a palavra chave function seguido do nome da função. O nome da função deve começar com uma letra ou o caracter '_' e o restante do nome pode conter os caracteres alfanuméricos e o caracter '_'. A forma geral de uma função é

```php
 function nome_da_função($arg_1,$arg_2,/*...,*/$arg_n){
   //Bloco de comandos
   echo"Exemplo de função.\n";return $valor_retornado;
 }
```

O comando **return** é opcional, já que não é obrigatório retornar um valor em funções no PHP, outra regra consiste em não permitir que sejam retornados múltiplos valores através deste comando. Para resolver essa necessidade, pode-se retornar listas e **arrays**.

No PHP as funções não precisam ser criadas antes de serem referenciadas, exceto quando uma função é definida condicionalmente ou quando uma função é definida dentro de outra função.

Exemplo

```php
<?php
$check = true;
/* Neste parte do programa, é possível fazer uma chamada das funções
g() e m(). Entretanto, as funções f() e n() não podemos ser chamadas
pois elas ainda não existem.*/
g();

// Função definida condicionalmente
if ($check) {
    function f(){
            echo "f() não existe até que o programa passe por aqui";
    }
}
// Função definida dentro de outra função
    function m(){
        echo "m() existe desde o começo do programa";

        function n(){
                echo "n() não existe até que m() seja chamada";
        }
    }
m();

// A partir deste ponto podemos chamar f() e m(). São funções do tipo
condicional e definida dentro de outra.
 function g(){
    echo "g() existe desde o começo do programa";
 }
?>
```

ARGUMENTOS

Informações podem ser passadas para funções através de argumentos. O PHP suporta a passagem de argumentos por **valor** ou por **referência**, também é permitido valores padrões de argumentos. Por padrão, as informações são passadas por **valor** para as funções. Assim, os parâmetros que a função recebe é tratado como variável local dentro do contexto da função. Exemplo de função com dois argumentos:

```php
<?php
  function f($curso,$disciplina){
    echo "Curso:$curso-disciplina:$disciplina";//aspas duplas
  }
  f("ADS","LinguagemdeProgramação");//passagem por valor
  f("ADS","EstruturadeDados");

?>
```

Para que um argumento seja passado por referência em uma função é preciso adicionar o caracter & antes do nome do argumento, na definição da função, exemplo:

```php
<?php
  function concatenandoStrings(&$string){
    $string.='Dia!';
  }
  $str='Bom';
  concatenandoStrings($str);//passagem por referenciadas
  echo $str;//saída:'BomDia!'

  ?>
```

No PHP, uma função também pode definir **valores padrão** no estilo C++ para argumentos escalares. O valor padrão **precisa ser uma expressão constante**, e não uma

variável ou uma chamada de função ou mesmo um membro de classe. Na declaração da funções, os argumentos que possuirem valores padrões devem vir após os argumentos sem padrões. Caso contrário, a função não funcionará como esperado.

```php
<?php
  function f($disciplina,$curso="ADS"){
    echo"Curso:$curso-disciplina:$disciplina";
  }
  echo f("BancodeDados");
  //Saída:Curso:ADS-disciplina:BancodeDados
  echo f("LinguagemdeProgramação",null);
  //Saída:Curso:-disciplina:LinguagemdeProgramação
  echo f("EstruturadeDados","Automação");
  //Saída:Curso:Automação-disciplina:EstruturadeDados
?>
```

VALOR DE RETORNO

Toda função pode opcionalmente retornar um valor. Esse valor é retornado pelo comando **return**. Exemplo:

```php
<?php
   function quadrado($num){
      return $num*$num;
  }
  echo quadrado(3);//Saída:'9'.
?>
```

Obs: o **return** é o resultado da função que será passado.

Ferramenta para testes

IDEONE.COM

Essa ferramenta pode ser encontrada no site **https://ideone.com/** . Na primeira janela que é mostrada você já pode inserir o seu código, selecionando antes a linguagem que será utilizada. Para o funcionamento correta desta ferramenta, basta clicar nos botões 1 e 2 da imagem abaixo, selecionando primeiramente a linguagem de programação e logo em seguida mandando rodar o código.