Fundamentos do PHP

Leonardo W. Sommariva

Sintaxe Padrão

Sintaxe Padrão

- A sintaxe delimitadora de scripts PHP inicia com '*<?php*' e termina com '*?>*' e pode ser escrita juntamente com códigos HTML. Por exemplo:

```
<h1>Seja bem vindo</h1>
<?php
      echo “Hello, World”;
?>
```

Sintaxe Padrão

- Além da sintaxe delimitadora padrão, é possível utilizar uma sintaxe alternativa conhecida como *shorts-tags.* Por exemplo:

```
<h1>Seja bem vindo</h1>
<?
        echo “Hello, World”;
?>
```

OBS: Para ativar as *shorts-tags* é necessário ativar a diretriz short_open_tag do PHP.

Declaração de Variáveis

Declaração de Variáveis

- Uma variável sempre começa com cifrão, $, seguido pelo nome da variável.

- As regras de nomenclatura de variáveis são:
  - Deve iniciar ou com uma letra ou com um *uderscore*;
  - Pode conter letras, números, *underscores* e outros caracteres ASCII;
  - Letras minusculas e maiusculas em nomes de variáveis são diferenciadas pelo PHP.

Declaração de Variáveis

- Exemplos de declaração de variáveis validas:
  - `$nome`
  - `$_nome`
  - `$_nome_completo`
  - `$nome2`
  - `$nome_2`

- Exemplos *case sensitive*:
  - `$nome`
  - `$Nome`
  - `$NOME`

As três variáveis acima são variáveis diferentes para o PHP.

Declaração de Variáveis

- A designação de valor para variáveis em PHP não foge do padrão utilizado pela maioria das linguagens de programação. Por exemplo:

  - `$cor = “azul”;`
  - `$numero = 21;`
  - `$idade = 30;`
  - `$soma = 30 + “21”;`

Escopo de Variáveis

Escopo de Variáveis

Variáveis podem ser declaradas em qualquer lugar em um script PHP. Mas o local de sua declaração influência em sua visibilidade no script. Em PHP as variáveis pode ter quatro escopos distintos:

- Variáveis Locais;
- Parâmetros de Função;
- Variáveis Globais;
- Variáveis Estáticas.

Escopo de Variáveis – Variáveis Locais

- Uma variável declarada dentro de uma função, será visível somente dentro desta função.

- Se dentro da função for declarada um variável cujo nome já é utilizado em outra variável fora da função, estas duas serão variáveis distintas.

Escopo de Variáveis – Variáveis Locais

```
$i = 2;

function xpto()
{
	$i = 5;
	echo “\$i dentro da função é $i”;
}

xpto();
echo “\$i fora da função é $i”;
```

- O saída do script acima será:

```
$i dentro da função é 5
$i fora da função é 2
```

Escopo de Variáveis – Parâmetros de Função

- Uma função que aceite parâmetros, deve possuir estes especificados entre parênteses logo após o nome da função.
- Qualquer parâmetro de uma função pode ser acessado e manipulado somente dentro da função. E este deixará de existir no momento em que a função for finalizada.

```
function muda_cor($nova_cor)
{
        $nova_cor = “azul”;
        return $nova_cor;
}
```

- O parâmetro *nova_cor* só poderá ser acessado e manipulado dentro da função *muda_cor.*

Escopo de Variáveis – Variáveis Globais

- Uma variável global não pode ser acessada diretamente dentro de uma função (Foi o que aconteceu no exemplo de "Variáveis Locais").
- Para acessar uma variável global dentro de uma função, é necessário utilizar a palavra-chave *global* na frente da variável que deve ser reconhecida como global. Por exemplo:

```
$var = 10;
function inc()
{
        global $var;
        $var++;
        echo "O valor de \$var é $var";
}
inc();
```

- O valor de *$var* será alterado para 11 dentro da função. E assim continuará mesmo ao término da função.

Escopo de Variáveis – Variáveis Globais

- Também se pode acessar variáveis globais em PHP através do $GLOBALS do PHP. O exemplo anterior ficaria assim:

```
$var = 10;
function inc()
{
        $GLOBALS[“var”]++;
        echo “O valor de \$GLOBALS[\“var\”] é ”.$GLOBALS[“var”];
}
inc();
echo “\n”;
echo “O valor de \$var é ”.$var;
```

- A saída do script acima será:
  - `O valor de $GLOBALS[“var”] é 11;`
  - `O valor de $var é 11;`

Escopo de Variáveis – Variáveis Estáticas

- Para definir um variável como estática, basta colocar a palavra-chave *static* na frente da declaração da variável.

- Diferente de uma variável local, uma variável estática não perde seu valor, mesmo quando a função é finalizada. Ou seja, quando a função for chamada novamente, o valor atribuído na ultima chamada da função ainda estará armazenado. Por exemplo:

Escopo de Variáveis – Variáveis Estáticas

```
function xpto()
{
        static $var = 0;
        $var++;
        echo "\$var == $var";
}
xpto();
xpto();
xpto();
```

- A saída do script acima será:
  - `$var == 1`
  - `$var == 2`
  - `$var == 3`
- Se a variável *$var* não fosse estática, a saída seria a seguinite;
  - `$var == 1`
  - `$var == 1`
  - `$var == 1`

Variáveis Superglobais

Variáveis Superglobais

- As variáveis superglobais são aquelas que podem ser acessadas em qualquer lugar de qualquer script. Alguns exemplos de variáveis superglobais do PHP são:
  - `$GLOBALS;`
  - `$_SERVER;`
  - `$_GET;`
  - `$_POST;`
  - `$_FILES;`
  - `$_COOKIE;`
  - `$_SESSION;`
  - `$_REQUEST;`
  - `$_ENV;`
- Para visualizar o conteúdo de uma variável global, basta chamar a função *printr* e passar a variável como parâmetro. Por Exemplo:

```
printr($_SERVER);
```

Estruturas de Controle

Estruturas de Controle

- Estruturas de controle determinam o fluxo do código durante sua execução, definindo se e quantas vezes determinado trecho de código será executado.

- Estruturas de controle se divide em duas:
  - Estruturas de seleção;
    - *if*, *else*, *elseif* e *switch*
  - Estruturas de repetição;
    - *while*, *do while*, *for*, *foreach*, *break*, *goto* e *continue*

if, *else*, *elseif* e *switch*

ESTRUTURAS DE SELEÇÃO

if

- A expressão *if* é uma das mais comuns entre as linguagens de programação. Sua sintaxe básica é:

```
if (<condição>)
{
        <código>
}
```

- Um exemplo pratico:

```
$numero = 123;
if ($_POST[“numero”] == $numero)
{
        echo “O número está correto”;
}
```

- Lembrando que as chaves só são necessárias caso o trecho de código submetido ao condicional tenha mais de uma linha. Caso contrário:

```
$numero = 123;
if ($_POST[“numero”] == $numero) echo “O número está correto”;
```

else

- A expressão *else* é uma alternativa ao *if,* ou seja, caso a condição falhar o código submetido ao *else* será executado. Sendo assim a expressão *else* pode ser considerada o esccape do *if,* e deve ser usada juntamente com este, por exemplo:

```
$numero = 123;
if ($_POST[“numero”] == $numero)
{
        echo “O número está correto”;
} else {
        echo “O número está incorreto”;
}
```

elseif

- A expressão *elseif* continua sendo um escape para a expressão *if,* porém ela também possui uma condição. Como por exemplo:

```
$numero = 123;
$aux = 12;
if ($_POST[“numero”] == $numero)
{
        echo “O número está correto”;
} elseif ($numer == $aux) {
        echo “O número está parcialmente correto”;
} else {
        echo “O número está incorreto”;
}
```

switch

- A expressão *switch* pode ser considerada uma variante da combinação *if-else*. Ela pode ser muito útil quando é necessário comparar uma variável com um grande número de valores. A sintaxe básica do *switch* é:

```
switch(<variável>) {
        case <valor>:
                <codigo>
                break;
        case <valor>:
                <codigo>
                break;
        default:
                <codigo>
}
```

switch

- Um exemplo prático:

```
$media = $_POST[“media”];
switch($media) {
        case 7:
                echo “Você foi aprovado”;
                break;
        case 8:
                echo “Você é um bom aluno”;
                break;
        case 9:
                echo “Você foi aprovado com mérito”;
                break;
        case 10:
                echo “Parabéns, você é um aluno exemplar”;
                break;
        default:
                echo “Você não foi aprovado”;
}
```

OBS: a expressão *default* representa a opção de escape, ou seja, se nenhum dos *cases* forem satisfeito, o código submetido ao default será executado.

while, do while, for, foreach, break, goto e *continue*

ESTRUTURAS DE REPETIÇÃO

while

- A expressão *while* necessita de no mínimo uma condição, e enquanto esta for satisfeita o *looping* será executado. A sintaxe padrão é:

```
while (<condição>)
{
        <código>
}
```

- A única restrição da cláusula *while* é que a condição precisa ser satisfeita para que o *looping* seja iniciado. Por exemplo:

```
$numero = 3;
$fatorial = 1;
while ($numero > 1)
{
        $fatorial *= $numero--;
}
echo “O fatorial é $fatorial”;
```

do while

- A expressão *do while* é muito similar a expressão *while.* A única diferença é que a condição não precisa estar satisfeita para o *looping* ser iniciado, pois a condição só é verificada ao final da execução do primeiro *loop.* A sintaxe básica do *do while* é:

```
do
{
        <código>
} while (<condição>);
```

- Pelo fato da condição só ser verificada no fim, o trecho de código submetido ao *do while* sempre será executado uma vez.

for

- A expressão *for* possui um mecanismo de *looping* mais complexo do que *while* e *do while.* A sintaxe básica é:

```
for (<exp1>; <exp2>; <exp3>)
{
        <código>
}
```

- Para a utilização da expressão *for*, é necessário entender suas expressões:
  - A primeira expressão (exp1), é avaliada na primeira repetição do loop;
  - A segunda expressão (exp2), é avaliada no começo de cada repetição;
  - A terceira expressão (exp3), é avaliada no final de cada repetição;

for

- Um exemplo da utilização da expressão *for:*

```
for ($real = 1; $real <= 5; $real++)
{
        echo “$real equivale a ”.($real*3.89)
                .”euros”;
}
```

- A saída será:

```
R$1.00 equivale a 3.89 euros
R$2.00 equivale a 7.78 euros
R$3.00 equivale a 11.67 euros
R$4.00 equivale a 15.56 euros
R$5.00 equivale a 19.45 euros
```

foreach

- A expressão *foreach* é exclusiva para arrays, pois em sua sintaxe básica é necessária a especificação de um array.

  - Para arrays sem chave:

```
foreach (<array> as $valor) {
        <código>
}
```

  - Para arrays com chave:

```
foreach (<array> as $chave => $valor) {
        <código>
}
```

OBS: A sintaxe para arrays sem chave pode ser utilizadas com arrays com chave também.

foreach

- Exemplo de utilização do *foreach:*

```
$array = array("um" => 1, "dois" => 2, "tres" => 3);

foreach ($array as $chave => $valor) {
      echo "A chave é ‘$chave’ e o valor é $valor\n";
}
```

- A saída será:
  - `A chave é ‘um’ e o valor é 1`
  - `A chave é ‘dois’ e o valor é 2`
  - `A chave é ‘tres’ e o valor é 3`

break

- A palavra-chave *break* finaliza a execução de um *looping.*

- Normalmente esta expressão é utilizada embutida em uma condição que necessita que o *looping* seja interrompido

goto

- A expressão *goto* faz com que o ponteiro da execução pule alguns comandos até o marcador especificado pelo programador. Por exemplo:

```
for ($i = 1; $i <= 10; $i++)
{
        if ($i == 9)
        {
                goto ponto;
        } else {
                echo “O número não é 9”;
        }

}
ponto:
        echo “O número é 9”;
```

continue

- A expressão *continue* tem uma certa semelhança com a expressão *break.* Porém, ao invés de interromper completamente o *looping,* somente é interrompida a interação atual do *loop* e é dado inicio a próxima interação. Por exemplo:

```
for ($i = 1; $i <= 10; $i++)
{
        if ($i < 9)
        {
                continue;
        } else {
                echo “O número 9 foi encontrado”;
        }

}
```

Referências bibliográficas

- GILMORE, Jason W. **Dominando PHP e MySQL.** 2011. Terceira Edição. Alta Books Editora. Rio de Janeiro.
- APACHE. **HTTP Server Project.** 2015. Disponível em <http://httpd.apache.org/>. Acessado em 3 ago. 2015.
- The PHP Group. **Manual do PHP.** 2015. Disponível em <http://php.net/manual/pt_BR/configuration.changes.php>. Acessado em 3 ago. 2015.