Estruturas de controle no PHP

Prof.: Alisson G. Chiquitto
chiquitto@unipar.br

Estruturas de controle

- “Em ciência da computação, estrutura de controle (ou fluxo de controle) refere-se à **ordem em que instruções**, expressões e chamadas de função **são executadas** ou avaliadas em programas de computador sob programação imperativa ou funcional.” (Wikipédia)

Estruturas de controle do PHP

- if
- else
- elseif/else if
- while
- do while
- for
- foreach
- break
- continue
- switch/case
- return
- require/include

if

```php
<?php
if ($expressao) {
    // bloco de instrucoes
}
```

- O IF permite a execução condicional de um bloco de instruções;
- A expressão será sempre avaliada como um boolean;
- Se a expressão for avaliada como TRUE, o bloco de instruções será executado;

if

```php
$a = 4;
$b = 3;
if ($a > $b) {
    echo "A é maior que B";
}
```

if

```php
$c = (5 > 6);
if ($c) {
    echo 'C é verdadeiro';
}

if ($d = (15 > 10)) {
    echo 'D é TRUE';
}
```

else

- Se a estrutura “if / elseif” não for executada, o “else” será executada;
- Deve ser utilizada logo depois do fechamento de um “if / elseif”;

```
if ($expressao) {
    // bloco de instrucoes
}
else {
    // executado se $expressao é FALSE
}
```

elseif / else if

- A combinação else + if;

```
if ($expr) {
    // instrucoes
}
else {
    if ($expr2) {
        // instrucoes
    }
    else {
        // instrucoes
    }
}
```

```
if ($expr) {
    // instrucoes
}
elseif ($expr2) {
    // instrucoes
}
else {
    // instrucoes
}
```

while

```
while($expr) {
    // instrucoes
}
```

- Loop mais simples do PHP;
- Executa o bloco de instruções enquanto $expr for verdadeira;

while

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

do while

```
do {
    // instrucoes
} while ($expr);
```

- Semelhante ao while, exceto que $expr será avaliado no final de cada iteração;
- Sempre faz no mínimo 1 iteração;

for

```
for ($expr1; $expr2; $expr3) {
    // instrucoes
}
```

- $expr1 é sempre executado no começo do for;
- Antes de cada iteração, $expr2 é avaliado: se Verdadeiro, executa o bloco de instruções, senão a execução do laço termina;
- Ao fim de cada iteração, $expr3 é executada;

for

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

foreach

- foreach fornece uma maneira fácil de iterar sobre arrays e objetos;
- Existem 2 sintaxes:

```php
foreach($array as $value) {
    // instrucoes
}

foreach($array as $key => $value) {
    // instrucoes
}
```

foreach

- Na primeira sintaxe, para cada iteração, o valor do item corrente é **atribuído** a $value, então o ponteiro interno do array avança uma posição;
- Na segunda sintaxe, além da atribuição de $value, a chave do item corrente é atribuida à $key;

foreach

```php
$array = array(
    'pais' => 'Brasil',
    'uf' => 26,
    'capital' => 'Brasilia',
);

foreach($array as $key => $value) {
    echo "A chave $key = $value <br>";
}
```

break

- A palavra-chave break **para a execução** das seguintes estruturas de controle:
  - while
  - do while
  - for
  - foreach
  - switch

break

```php
$i = 1;
while ($i < 10) {
    if ($i == 5) {
        break;
    }
    $i++;
}
```

continue

- continue é utilizado dentro de laços;
- é utilizado quando é necessário ignorar o **resto da iteração atual**;

```
$i = 0;
while ($i < 10) {
    $i++;
    if (($i % 2) != 0) {
        continue;
    }
    echo $i;
}
```

switch/case

- A estrutura de controle switch é similar a séries de if/elseif/else;
- Útil para comparar a mesma variável com diferentes valores;

switch/case

```php
$i = 3;
switch ($i) {
    case 0:
        echo 'i == 0';
        break;
    case 1:
        echo 'i == 1';
        break;
    case 2:
        echo 'i == 2';
        break;
    case 3:
    case 4:
        echo 'i == 3 ou 4';
        break;
    default :
        echo "i == $i";
}
```

return

```php
function soma($a, $b) {
    return $a + $b;
}
```

- Se utilizado dentro de uma função, a instrução “return” para a execução da função corrente e retorna um valor;

include

- Inclui um arquivo para ser executado;
- Funciona como um Ctrl+C / Ctrl+V, mas em tempo de execução;

```php
include 'funcoes.php';

echo soma(4,2);
```

require

```php
require 'funcoes.php';
echo soma(4,2);
```

- Semelhante a instrução “include”;
- A diferença é quando o arquivo a ser incluído **não existe**:
  - O “include” dispara um erro, mas o script **continua** sendo executado;
  - O “require” dispara um erro, mas **para a execução** do script;